2018

Evolving Encoded Marking for Glove-Based Pose and Posture Recognition

Joshua V. Gyllinsky
University of Rhode Island, jgyllinsky@my.uri.edu

EVOLVING ENCODED MARKING FOR GLOVE-BASED POSE AND POSTURE RECOGNITION

BY

JOSHUA V. GYLLINSKY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2018

MASTER OF SCIENCE THESIS

OF

JOSHUA V. GYLLINSKY

APPROVED:

Thesis Committee:

Major Professor Jean-Yves Hervé

Gérard M. Baudet

Lenore M. Martin

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2018

ABSTRACT

The feasibility of encoding pose and posture information within surface textures for digital reconstruction of the inverse-mapping problem of computer vision was explored. This was done using a combination of three biology-inspired artificial intelligence techniques. Initial steps were conducted in this pursuit and the path to further research is laid out. It was found that textures can be used to encode information on a 3D surface to supplement computer vision systems.

ACKNOWLEDGMENTS

In accordance with the University of Rhode Island Graduate Student Handbook and the thesis requirements for graduation, the following is presented:

I would like to acknowledge my advisor, whose likely unprecedented knowledge, guidance and mentorship cannot be measured. It is about going "beyond advisable", moving the boundaries of knowledge and making the world better.

Similarly, I would also like to acknowledge my committee for their time reviewing my work.

The support of family and friends were critical during this endeavor and are sincerely appreciated.

DEDICATION

This work is dedicated to free speech and the open standards community.

PREFACE

More information about the structure of the document can be found in Appendix C.1 and Appendix C.2 for the LibHand citation.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure **Page**

CHAPTER 1

Introduction

1.1 Statement of the Problem

The initial prompt of this project was to research the question of whether it is possible to reliably encode static information on the surface of an articulated non-stationary 3D object such that its projection can be distinct enough to infer state information of the 3D object for local regions. The objective direction of this academic work was aimed to explore, and perhaps discover if, and to what extent, information could be reliably encoded on the surface of a digital deformable 3D model of a gloved hand, and then read by a discrete sensor representative of a digital camera, for use in a hand recognition system. It was hypothesized that this solution, if then produced, might provide an accurate and precise representation of the hand, and perhaps it could be comparable with alternative gloved solutions which often require custom hardware or sensors other than a commodity digital camera.

1.2 Project Goals

This project represents an initial exploration of glove-based hand pose and posture identification with supplemental visual queues, or encoded markings. The overarching goals of this academic endeavor were to learn about machine learning techniques, computer vision, and do so through an initial exploration of a concise problem. It is not the intention of this project to provide a final compendium on the topic, nor is it to produce absolute metrics. Rather, through exploratory programming and analysis, only perhaps to discover some insights about the problem.

1.3 High-Level Structure

The general structure of the designed solution space searching program is a composite construction of competing optimization systems. Their interaction is similar to an iterative game. An underling premise is that they can search the solution space without necessarily being destructive of each other, and moreover, if set up in a balanced way, actually improve each other through pressured competition.

Evolutionary computing (EC) software based on Genetic Programming (GP) is written and executed to assist in semi-supervised training of a Neural Network (NN), which itself is used to identify patterns generated by transformations and occlusions of 3D surfaces with the assistance of textures as generated by Genetic Algorithms (GA). The generated systems are compared, curated and selected continuously during the project. The resulting end product's accuracy and precision for representation of the hand is evaluated, and whether it is comparable with existing alternative gloved solutions is explored.

One way of encoding information into an image is through the use of a barcode. There are many proposed barcode standards, both using 1D and 2D encoding methods. One of the most commonly used 2D barcode is known as a "QR Code". QR Codes are often used for encoding URL links. Figure 1 shows an example of how such a 2D barcode could encode information of transformations of a model of a hand. This encoding could be mapped onto a glove, as shown in Figure 2.

Figure 1. Encoded data in 2D Image (QR Code).

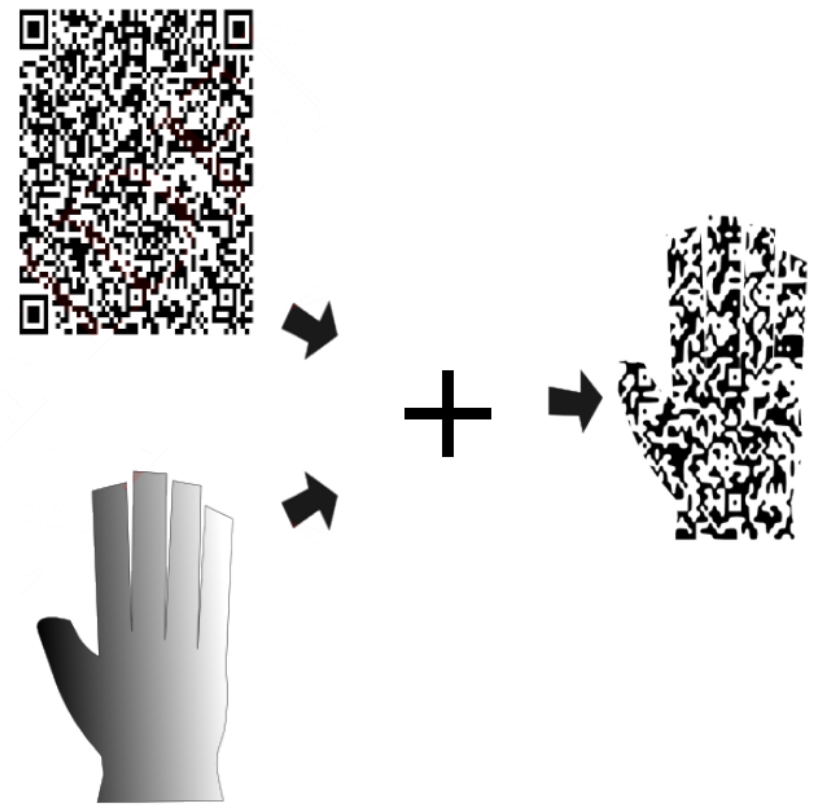


Figure 2. Conceptual representation of encoding information on a 2D shape of a hand.

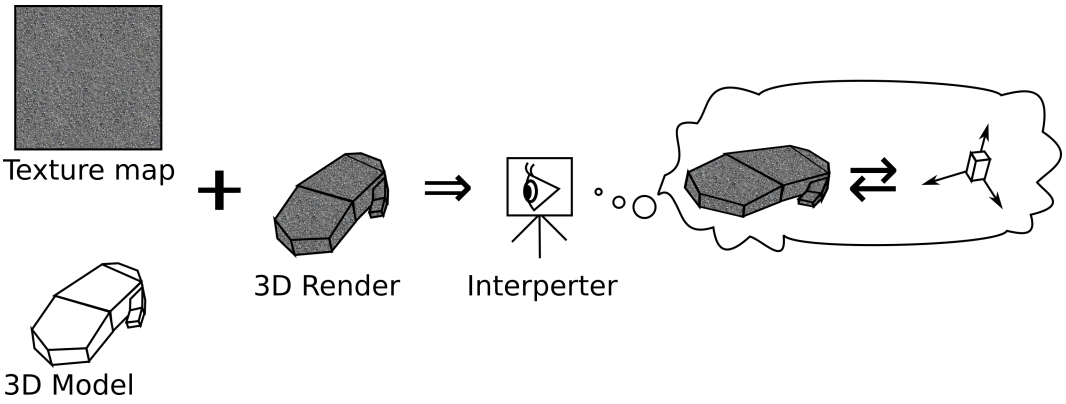


Figure 3. Conceptual process of encoding information on a 2D shape of a hand.

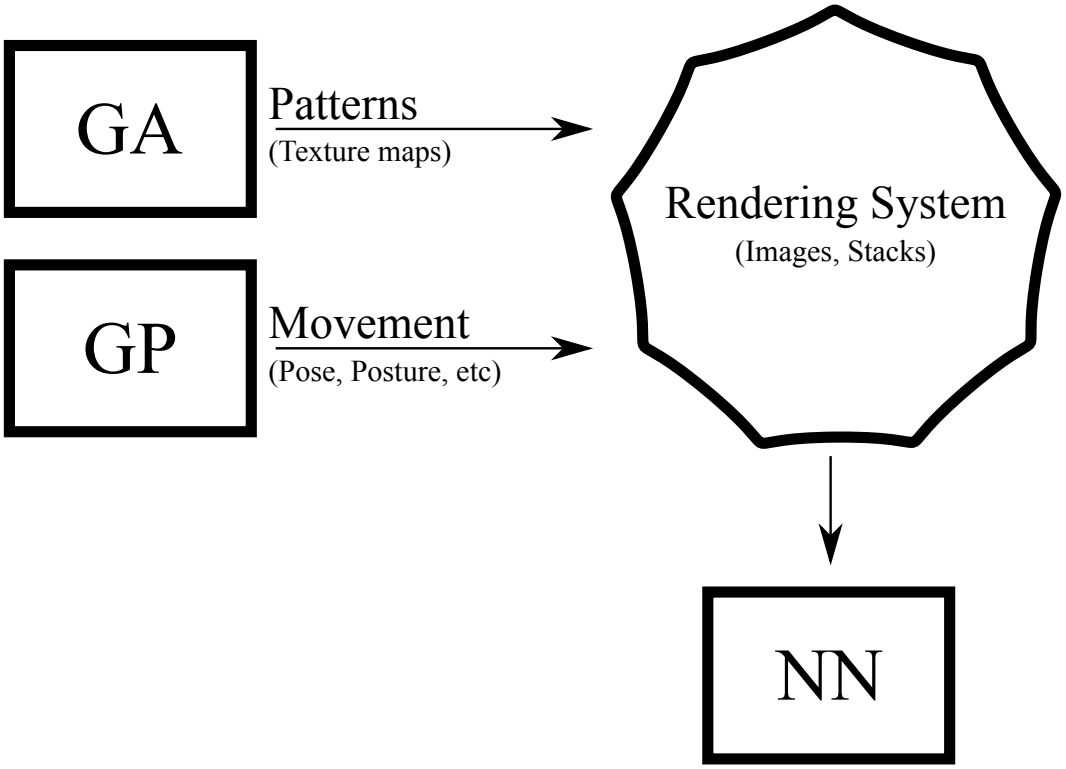


Figure 4. Conceptual possible high level design

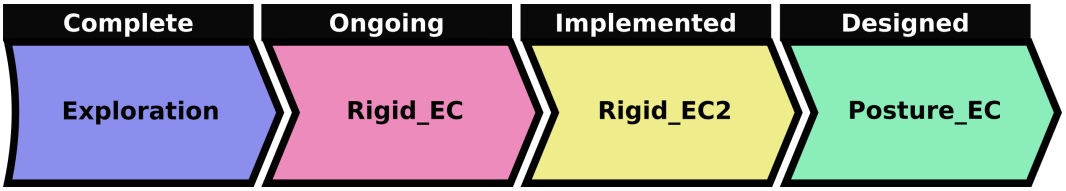


Figure 5. Systems

The glove being designed requires neither a power source nor affixed wires, aside from those needed by the processing computer and digital camera. Being battery-less, the proposed solution would enable a user to pick up a glove and use it without concern of its current power state. This project proposes a platform for glove-based hand gesture recognition with minimal computational overhead during use. During the project a novel hierarchal AI system was designed and implemented. It expanded the current knowledge in the subject, and has practical applications. The proposed system not only uses inexpensive hardware but also greatly reduces computational load needed to recognize the glove's transformational data, namely pose and posture. The computational load reduction was accomplished having pre-processed data allowing the model to internalize enough information about the problem. The one-time discovery of a texture map suitable for use on the 3D model and a means of reading the texture map was processed on a personal computer. However, the system has been designed to eventually run on a High-Powered Computing (HPC) cluster. Moreover, the selected types of AI that were used were chosen because of their suitability for parallelization, allowing for scalability as different computational resources become available. It was an objective of this work that this system would be a suitable starting place for further development resulting in later deployment as an input system for use with low-power, low-resource, real-time embedded devices. While this proposed system is designed to perform hand posture recognition, the recognition platform may be extensible to any object. However, that exploration is outside the scope of this study.

There are practical applications for a fine-tuned system that can reliably transpose hand postures into virtual manipulables. As such, there are currently numerous commercially available solutions marketed and actively being adopted in the

consumer market. However, no known current technology offers the compromises and benefits offered by this idea. The proposed non-powered glove solution is unconstrained by wires, leaving the user freedom of mobility without a fixed armature. This means deployment is relatively simple and fast, requiring only a computer, a camera, software based on this study's findings, and a user wearing the glove.

The system architecture consisted of GA, GP, NN components from the highest view. Shown in Figure 12 is a high-level overview of the project's components. The result of such a system is a looped flow, where a single iteration from texture map to resulting description is shown in Figure 11.

Moreover, the types of AI that were used were chosen because of their suitable for parallelization, allowing for scalability as different computational resources become available. It was an objective of this work that this system would be suitable starting place for further development resulting in later deployment as an input system for use with low-power, low-resource, real-time embedded devices. While this proposed system is designed to perform hand posture recognition, the recognition platform may be extensible to any object. However, that exploration is outside the scope of this study.

There are practical applications for a fine-tuned system that can reliably transpose hand posture into virtual manipulables. As such, there are currently numerous commercially available solutions marketed and actively being adopted in the consumer market. However, no current technology offers the compromises and benefits offered by this idea. The proposed non-powered glove solution is unconstrained by wires, leaving the user freedom of mobility without a fixed armature. This means deployment is relatively simple and fast, requiring only a computer, a camera, software based on this study's findings, and a user wearing the glove.

1.4 Terminology and Commentary

1.4.1 General Terminology

For the purposes of this work and to aid understandability, the changes in the 3D model caused by its rotation, translation, and scaling, as well as its deformations caused by either lattice deformations or bone deformations, will be collectively referred to as the object's transformations. Changes in the 3D model not including deformations will be referred to as affine transformations. A texture map is an image that can be wrapped onto a 3D object to add the appearance of a material to the polygons making up the object. When referring to the three-dimensional object, usually when rendered with associated texture map and armature (bone skeleton), the term 3D Model will be used. This should not be confused with a model which might refer to any simplified representation, usually of on a computer, of a much more complex system, usually existing in the real world. The distinction between a subsystem and a system is whether the item in question is being described in the context of a larger, encapsulating system. Individual parts of a system which can function as standalone objects, are modules and, a similar distinction is made for modules and submodules.

1.4.2 Machine Learning Terminology

While working on this thesis, it became common to refer to Artificial Neural Networks (ANNs) as simply Neural Networks (NNs), and to specify when the NN implementation was that of Deep Learning (DL). The NN system used at the beginning of this thesis was a classical ANN, but later it was replaced with a DL system. Language changes overtime, and so ANN remains in-place for where the classical methodology was used, and NN is used generically for where DL was used and, for consistency, the overall system. The original implementation had the GP and GA system separately implemented, even though the code for the GP

system was based on that of the GA system. This collectively became the EC subsystem, which in turn was made of both the GA and GP subsystems. This lead to the naming convention used for the Rigid_EC, Rigid_EC2, and Posture_EC systems. The names of individual programs, which were singular snapshots of the changing subsystems, are prefixed with their respective subsystem. For example, "nn_StaticPosture" and "rs_StaticRenderer", belong to the NN and RS subsystems, respectively, and both are submodules of the Exploration system.

1.4.3 Pose and Posture Terminology

It should be noted that some of the tools in the ecosystem used for this project may use slightly different nomenclature. For example, a prominent software library used in the implementation of this thesis project for rendering hand postures, LibHand, does not make a distinction between "poses" and "postures". Instead, in that library all deformation configurations are simply referred to as "poses". This interpretation can, with a slight distinction, be fit within the terminology used here if it is understood that within LibHand the resulting model is treated as a different one than the source model. In this way, with regard to the classifier system, which only classifies amongst specific postures, these may be referred to in the context of LibHand's concept of "poses", as "LibHand Poses", but will be treated as "postures" going forward. When deformations are actively taking place, "postures" will always be used.

CHAPTER 2

Review of the Literature

2.1 Background

2.1.1 Machine Learning Techniques

Evolutionary Computing

Evolutionary Computing (EC)-based Machine Learning (ML) techniques are clustered due to their attempt to perform their optimization search in a way conceptually similar to the various processes biological organisms use. The name however might be a misnomer, in that not all biologically inspired techniques fall in this category, and some which are focused on population-based search often employing Swarm Intelligence (SI) using biology inspired techniques are. Search methods such as the various Colony Optimization algorithms, including Ant Colony Optimization (ACO) algorithms and Artificial Bee Colony algorithms, are examples.

Neural Networks are loosely inspired by neurons and well are vastly simplified, appear to work relatively well.

A cartoon illustration demonstrating a single generation of a population in a GP system is shown in Figure 6. A cartoon illustration of recombination as part of GA is shown in Figure 7. The overall flow of a GA system from an individual perspective is shown in Figure 8. Genetic algorithms have been used for many things, including optimization of molecular modeling [1]. Genetic Programming (GP) is a type of AI notable for its use of trees as the virtual genetic material used to search a solution space.

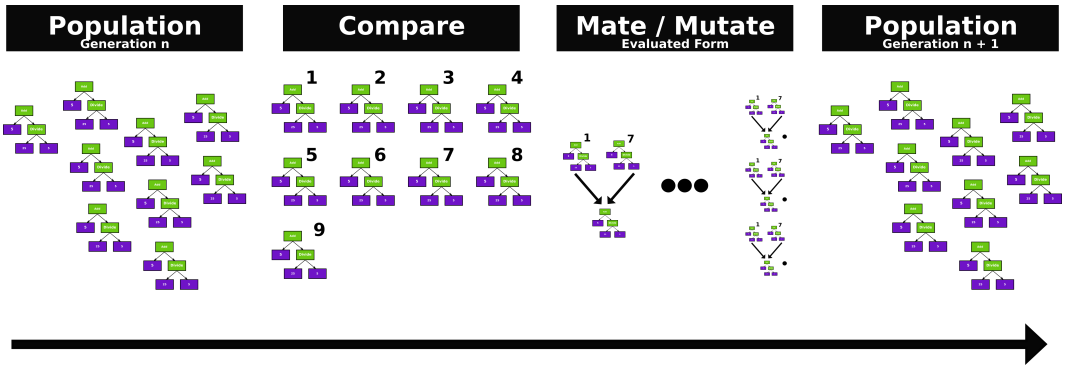


Figure 6. Generation of a population in an evolutionary computing program.

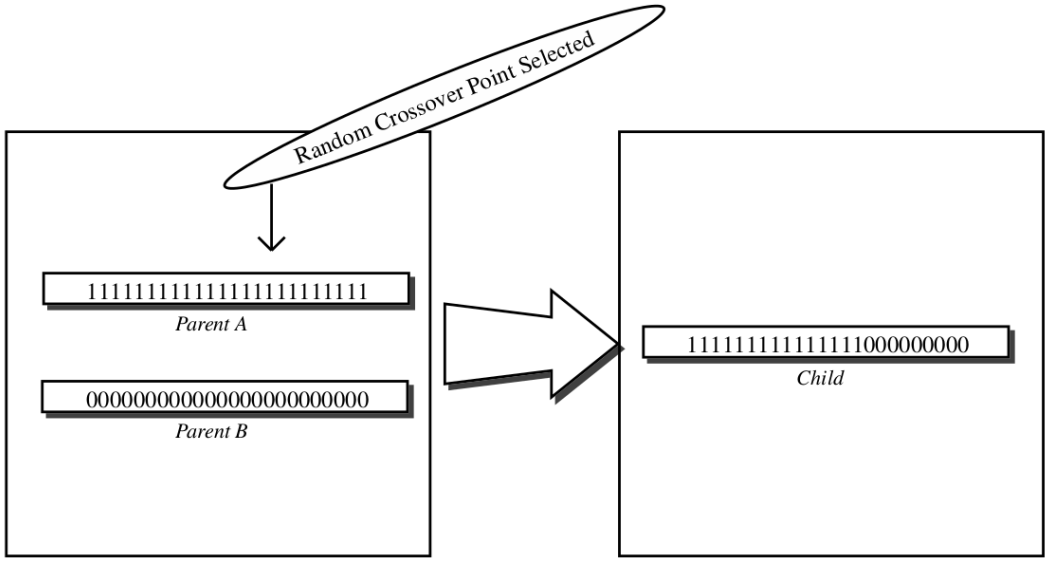


Figure 7. Genetic Algorithm: Recombination.

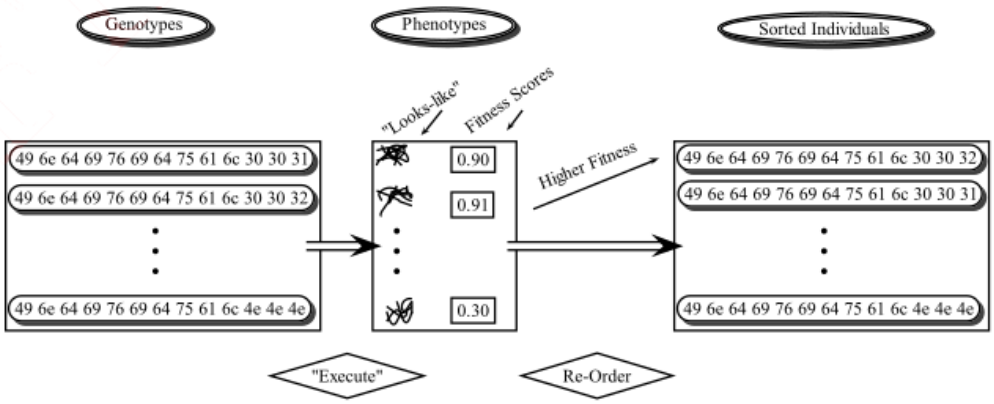


Figure 8. Overview of GA

2.1.2 Computer Vision

Classical Computer Vision and Image Processing

With the field starting arguably in the 1950's, computer vision researchers have long hoped for a solution with the robustness of the human visual system. Classical computer vision (C-CV) traditionally centers around explicit modeling of the physical world. The images are treated as matrices and these types of operations are used on them. From this data and image processing, the computer generates a precept, or internal representation of its physical space. This does not often lead to generalized models, but has been proven to be successful in isolated domains. Despite C-CV's inherent flaws, such as overly simplistic modeling, unnecessarily complex mathematics, and the overtly limited applicability for each individualized solution, it is a well established and is still widely used. Fortunately, machine learning and computer hardware architecture has come to the rescue in the last couple years.

Current Computer Vision

The promise of modern computer vision techniques such as those presented with deep learning, is that robust visual classification and analysis systems can be

developed, tested, and implemented with minimal direct programming.

Image Comparison Techniques

Image comparison techniques exist within the intersection of the fields of computer vision and image processing. These are non-trivial problems where extensive research has gone into the development of a plethora of alternative solutions. This is an active field, whose results can be seen on consumer-level products such as cameras which might be able to detect faces in real time, to advanced techniques such a stitching photos together by looking at common structural information as used in the field of photogrammetry.

One relatively early computer vision technique for object identification is HOG, and is described in a 1980s patent [2]. General regions of a source image are examined for a metric of "interest" that could at a later point be compared against it. This system however has low specificity and is not necessarily well resistant to three-dimensional transformations of the source image which result in large changes in form.

For the task of feature identification, two noteworthy algorithms are SIFT and ORB. The former is however proprietary, with some exceptions for research, while the latter is open source. These allow for identification and, across sets of images, tracking of features of interest. To some extent this allows for identifying the distance between two images as a binary, whether or not a feature of interest exist between two inputs.

Another task in this field is that of determining a scaled distance between two images. Algorithms of this type provide metrics allowing for the determination of similarity ranking between input images. SSIM is an algorithm in this category. The advantage of SSIM over some more simple techniques is its invariance to rescaling of the source images and resistance to some types of common noises such

as salt-and-pepper graininess.

Textures

Special textures generated by machine learning techniques have also been used to fool other machine learning computer vision systems [3]. Some of the more recent of these types work by using generative adversarial neural networks. The configuration is comparable, but the outcome, as caused by the fitness function, is very different. Effectively, these systems are structured such that the generative subsystem generates texture encodings based on feedback from the critic system, where instead of rewarding the correctly identified classification, the wrong categorization is promoted. Thus when the object is presented to a classifier similar to that of the critic system's, the object has a higher likelihood of being misclassified. The result is a type of camouflage where even if a human would perceive the correct classification, a computer vision classifier which may under other contexts achieve super human visual prowess, would be fooled.

2.1.3 Modeling the Human Hand

Significant research has gone into identifying the best ways to model the human hand.

While there may be high variation amongst individuals, the greatest part of the difficulty is often deciding on the model's degrees of freedom (DoF).

Figure 9 demonstrates how angles of a 3D model's armature could represent a hand's posture information.

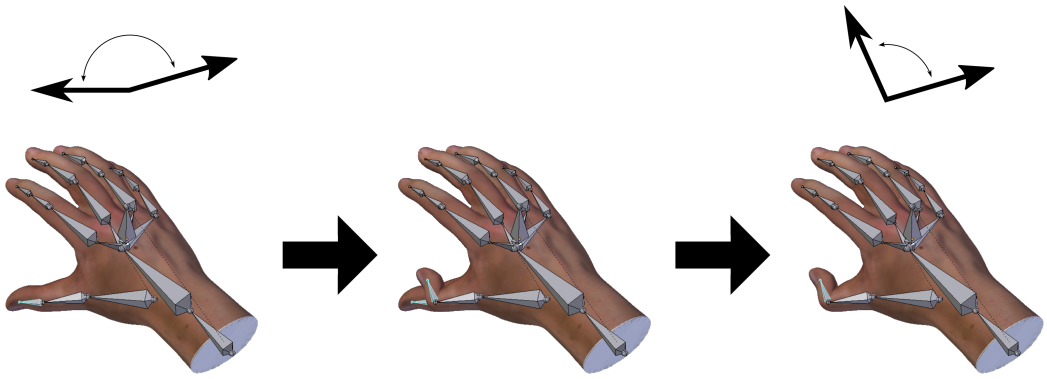


Figure 9. Hand Bones

2.1.4 Dimensionality

Dimensionality of the data necessary to reproduce a pose or posture is directly tied to the model's DoF. Dimensional reduction is therefore a necessity if the model's complexity surpasses the capabilities of the data processing tools available.

Communication of data with large dimensionality can pose difficulties as images, paper, and even standard computer screens are two-dimensional.

2.2 Related Works

2.2.1 Human-Computer Interfaces

One of the primary nonverbal communication tools that humans use are hand gestures. As part of body language this is a well-studied topic. Hand gestures are the movements of hands over time. Pose and posture refer to the instantaneous configuration. To describe the hands position, one would use pose (eg. holding one's hand up), and posture refers to the internal shape, or the deformation of the hand (eg. bending a finger making a pointing shape).

Significant work has already been done on hand gesture recognition systems [4, 5, 6, 7]. In fact, over the last several years, many types of solutions have been proposed. These proposed solutions to this problem have been sorted into several clusters, based on the techniques, sensors, and interaction methods used.

The academic field of Human-Computer Interaction (HCI) studies these modalities. While there are many excellent competing systems now available there does not seem to be a complete solution and rather depending on the domain of activity, one must choose [8].

At the price of reducing depth control, another way of achieving hand-based interaction is coupling the interaction to a surface. Examples of these types of interfaces are touch screens, and various interfaces have been discussed and examined previously [9]. Common hardware for achieving this are touch sensitive surfaces, such as those operating with detection from sensors for surface acoustic waves, resistance, and capacitance.

Both external and internal, back-positioned, visual detectors have also been proposed for this task. Various methods of using infrared sensors have been used to detect user interaction in this way.

2.2.2 Glove Systems

As can be seen in Figure 10, there are several glove systems available and four of them are highlighted here in this cartoon rendering. A "data-glove" such as those produced by Reza, Gyllinsky, *et al.* is an alternative type of glove [10]. This type of "data-glove" is also known as an e-textile, and specifically is used to for assessing the wearer's body poses, posture, or gestures. Similar systems have also been proposed for the face, feet and full body recognition [11].

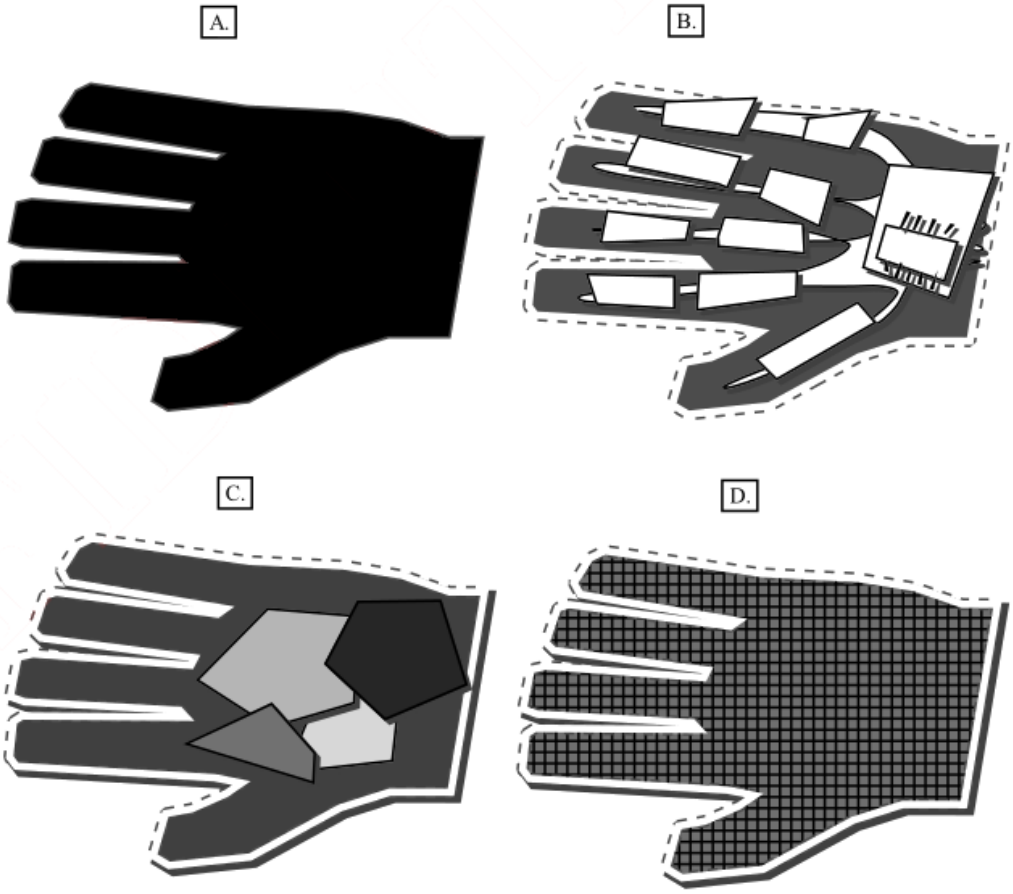


Figure 10. Four example techniques for hand posture capture. A: bare hand, this might be computer vision, B: data glove, C: simple symbols as markers, D: patterns

Numerous computer vision-based human recognition systems already exist and are being developed [7]. Some of these focus on the overall body language, or on the face, or hands and others somewhere in-between. Hand gesture recognition is an open problem in computer vision with many alternative solutions, each with its own advantages. Hand gesture recognition systems can be divided into several categories such as whether the system is gloved or non-gloved, powered or non-powered, or computer vision based or not. A system might focus on identifying specific translations and rotations, which will be referred to as poses, others include deformations in addition to pose information, which will be referred to as

postures, while others may include time, backward and forward examination of interpretations, and relaxation information, such as necessary for gestures. There is a disadvantage in using glove-based gesture and pose recognition strategies from the user's perspective, which is the need of the gloves themselves. A glove is a hand covering textile that, when used for computer vision, often includes sensors or other systems to improve hand posture recognition speed, accuracy, and precision. Bare-hand recognition approaches address this disadvantage, but currently lack the accuracy and precision required in many use-cases.

Most implementations of gesture recognition require special sensors and equipment. Wireless gloves allow significant freedom of movement, but they often require a power source. This means replacing the fuel or power physically or frequently recharging batteries over the glove's lifetime. A non-powered glove has significant advantages, such as accessibility due to low entry cost, limited maintenance, lack of a recurring power expense for the glove itself, and minimal preparation before use.

Several non-powered gloves with special markings for use as part of a computer vision system have already been proposed [7, 12, 13].

List of References

[1] B. Ott, J. V. Gyllinsky, N. Jouett, H. Alashwal, and L. M. Martin, "Parameter optimization using genetic algorithms for namd - saving time on computational chemistry with an initial ga pre-processing step," 2017. Accessed on April 15, 2018. Available: https://web.uri.edu/gradconference/

[2] R. McConnell, "Method of and apparatus for pattern recognition," Jan 1986, uS Patent 4,567,610. Accessed on April 15, 2018. Available: http://www.google.co.uk/patents/US4567610

[3] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," *CoRR*, vol. abs/1707.07397, 2017. Accessed on April 15, 2018. Available: http://arxiv.org/abs/1707.07397

[4] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, 2007, special Issue on Vision for Human-Computer Interaction. Accessed on April 15, 2018. Available: http://www.sciencedirect.com/science/article/pii/S1077314206002281

[5] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, Jul 1997.

[6] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *IEEE Computer Graphics and Applications*, vol. 14, no. 1, pp. 30–39, Jan 1994.

[7] F. Lathuilière and J. Y. Hervé, "Visual tracking of hand posture with occlusion handling," in *Proceedings of the International Conference on Pattern Recognition*, vol. 3, Barcelona, Spain, 2000, pp. 1129–1133 vol.3.

[8] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, Feb. 2011. Accessed on April 15, 2018. Available: http://doi.acm.org/10.1145/1897816.1897838

[9] M. Wu and R. Balakrishnan, "Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays," in *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '03. New York, NY, USA: ACM, 2003, pp. 193–202. Accessed on April 15, 2018. Available: http://doi.acm.org/10.1145/964696.964718

[10] M. Abtahi, J. V. Gyllinsky, B. Paesang, S. Barlow, M. Constant, N. Gomes, O. Tully, S. E. D'Andrea, and K. Mankodiya, "Magicsox: An e-textile iot system to quantify gait abnormalities," *Smart Health*, 2017. Accessed on April 15, 2018. Available: http://www.sciencedirect.com/science/article/pii/S2352648317300296

[11] M. Abtahi, N. P. Constant, J. V. Gyllinsky, B. Paesang, S. E. D'Andrea, U. Akbar, and K. Mankodiya, "Wearup: Wearable e-textiles for telemedicine intervention of movement disorders," in *Wearable Technology in Medicine and Health Care*. Elsevier, 2018, [Expected publish date: June 1st, 2018]. Accessed on April 15, 2018. Available: https://www.elsevier.com/books/wearable-technology-in-medicine-and-health-care/tong/978-0-12-811810-8

[12] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," in *Proceedings of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 63:1–63:8. Accessed on April 15, 2018. Available: http://doi.acm.org/10.1145/1576246.1531369

[13] V. F. Pamplona, L. A. F. Fernandes, J. ao Prauchner, L. P. Nedel, and M. M. Oliveira, "The image-based data glove," in *Proceedings of the X Symposium on Virtual Reality*, SBC. Porto Alegre, RS: SBC, 2008, pp. 204–211.

CHAPTER 3

Methodology

3.1 High-Level Goal

If the high-level project objective is that of Figure 3 in Chapter 1, then Figure 11 shows this objective as part of a implementable system. Figure 12 shows the module structure of the final system.

The final design had the following uses for the GA, GP, and NN submodules, as shown in Figure 12 and implemented through the development timeline as shown in Figure 14. GA was used to represent the texture, GP was used to represent changes to the 3D mesh, such as skeletal armature information, and NN was used to generate the reader program (RP) and to provide information for the scoring system's fitness evaluation (FE) via its calculated indicated accuracy per validation cycle, having been trained on the 3D renders (3DR) output of the rendering system (RS).

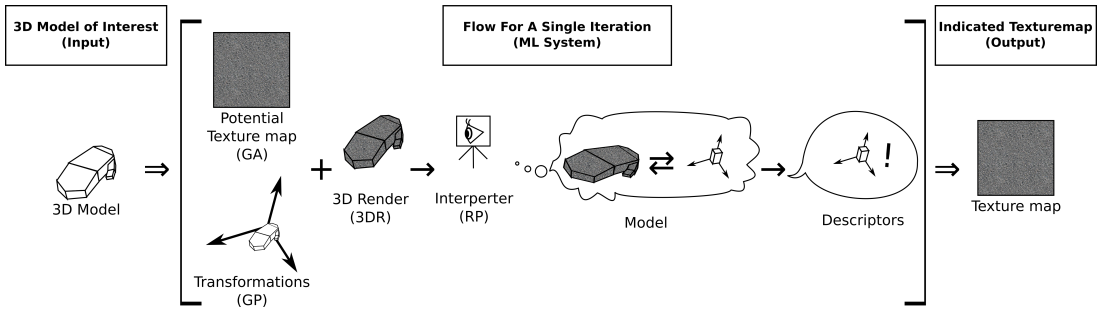


Figure 11. Overview of flow from texture map to descriptor

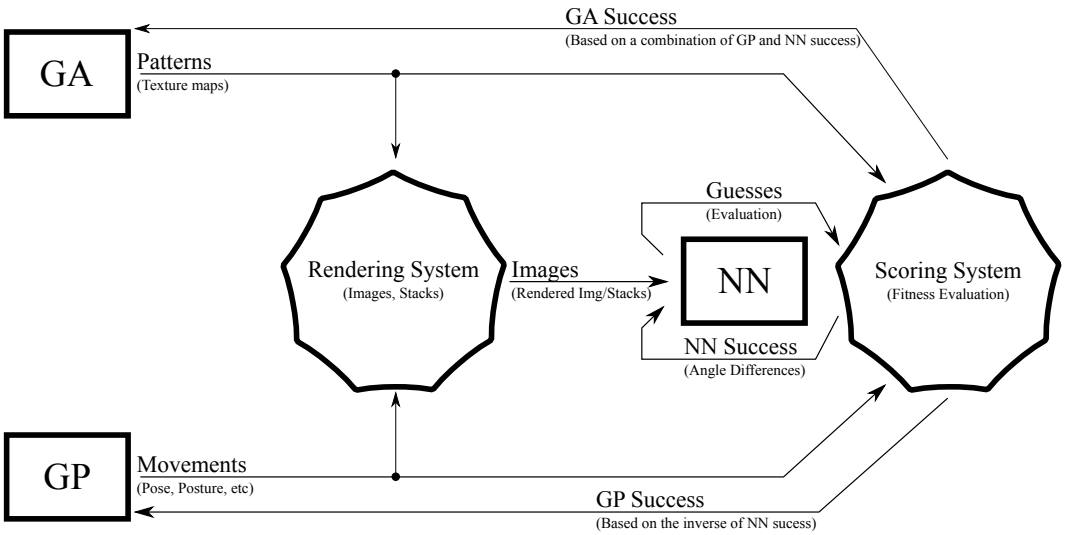


Figure 12. Overview of modules

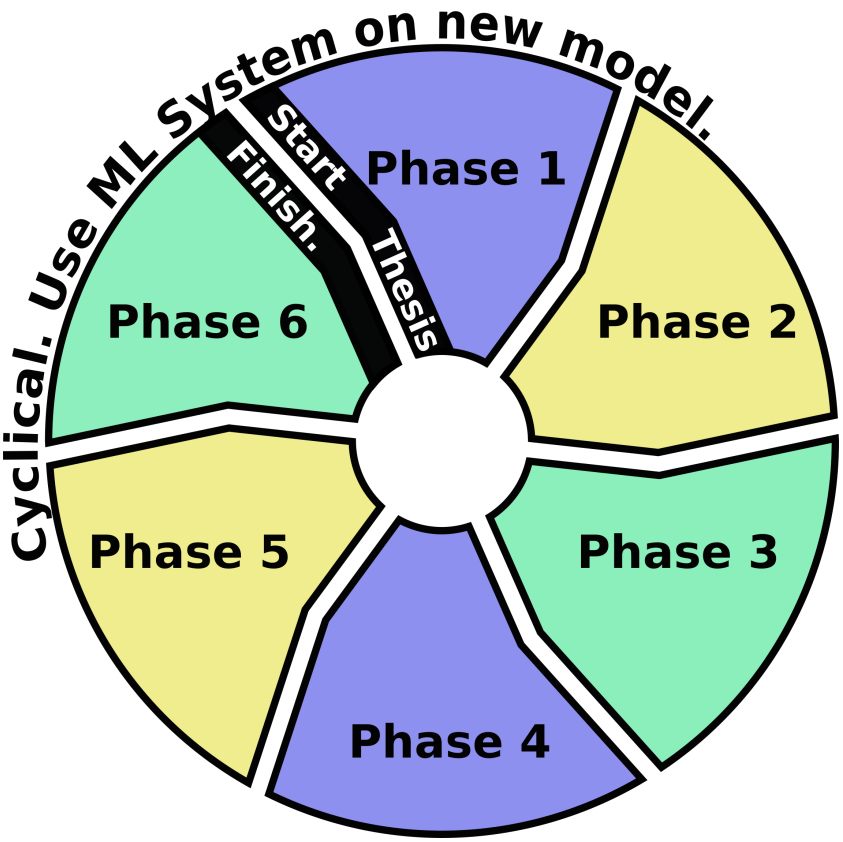


Figure 13. Intended Phases

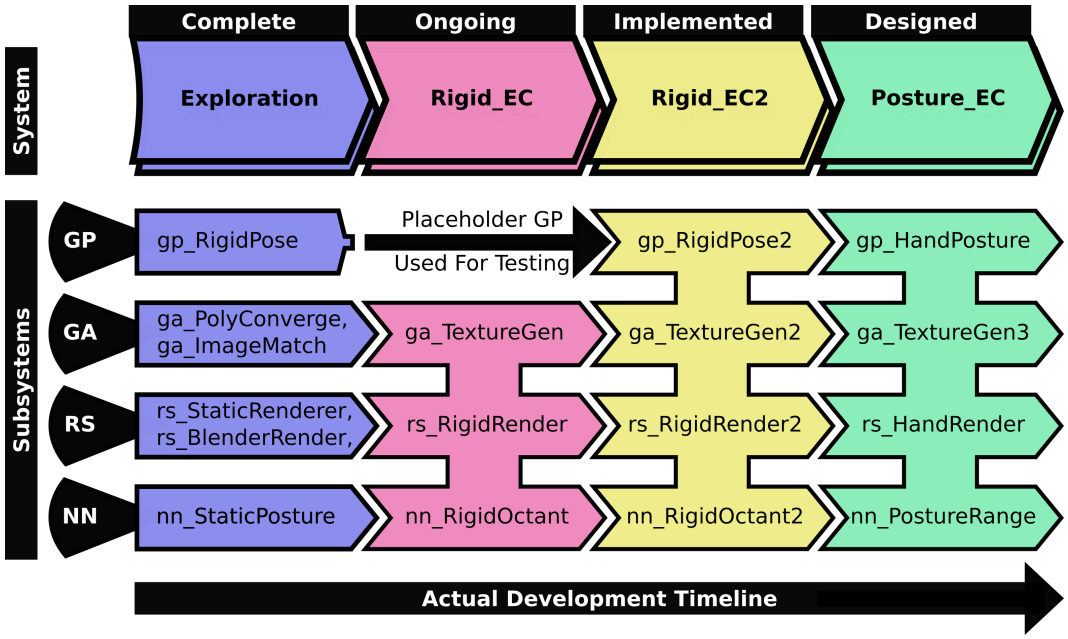


Figure 14. Systems and Programs

3.2 Phases

Although the implementation of the ML Systems did not match that of the original proposed phases structure, using the organization of phases matches the design process in-so-far as sub-objectives. For this reason, and because it was the intended structure at the projects outset, it is used here. However, specific results are called out by their respective overall system. Figure 13 illustrates the cyclical nature of the intended phases design, where the cycle executed for the initial project but could also be repeated for the creation of further ML System implementations on new models. Figure 14 documents the implemented systems and their status at the time of this writing.

3.2.1 Phase 1: Numerical Measurement Methods

To take advantage of machine learning to assist in solving this problem, it was necessary to have clear criteria for determining if solutions were closer or further to an optimal solution. A way of calculating such a numerical measurement, or

fitness score, for possible solutions needed to be determined. Early in the project it was hypothesized that four types of comparisons might need to be evaluated for establishing a solution's fitness. These criteria were thought to provide a reproducible way of determining a potential solution's fitness. These candidate metrics, shown below, needed to be tested empirically during the project as well.

1. A criterion for quantifying the differences between texture maps.
2. A criterion for quantifying the differences between different poses of the same 3D model.
3. A criterion for quantifying the differences between different postures of the same 3D model.
4. A criterion for quantifying the dissimilarities between rendered images of the textured model with transformations applied.

It was believed that these comparisons are possible and different applicable techniques were evaluated. The texture maps could be compared using techniques such as comparing the image's histograms, or by computing some cumulated difference of the intensity values over a region of interest (SSD [1], SSIM [2, 3, 4, 1]).

To compare two postures, the composing bone angles could be compared using dot products. To compare two poses, the root bone of the armatures could be positioned at the origin such that the pose information could likewise be compared using dot products. To compare differences of rendered images, methods similar to those used for texture maps may also be applicable. One major difference between the two problems is that the difference between the input image and the model-generated image is only relevant in parts of the image that correspond to the hand. This means that, prior to applying the difference criterion, one must perform a segmentation task to isolate the hand in the input image. Segmentation

is a classical problem in computer vision, concerned with isolation of sub-image content [5, 1]. This is a difficult problem outside of special laboratory conditions, due to noise and lack of ground-truth.

Two different types of hand posture tools are proposed here. The first one is more advanced and more complicated where angles of the model are predicted; in the second one, specific poses/postures are identified.

Angles as Categories

One possible method for encoding the angles would be using one hot encoding. This means establishing that, for every possible angle that the classifier could identify, the angle itself would count as a class.

Poses as Categories

Specific hand poses were selected to represent categories in a classification problem because this use of the CNN architecture is well documented. At the time of this writing, LibHand provides eight example postures as part of the software package. These default postures are shown in Figure 15.

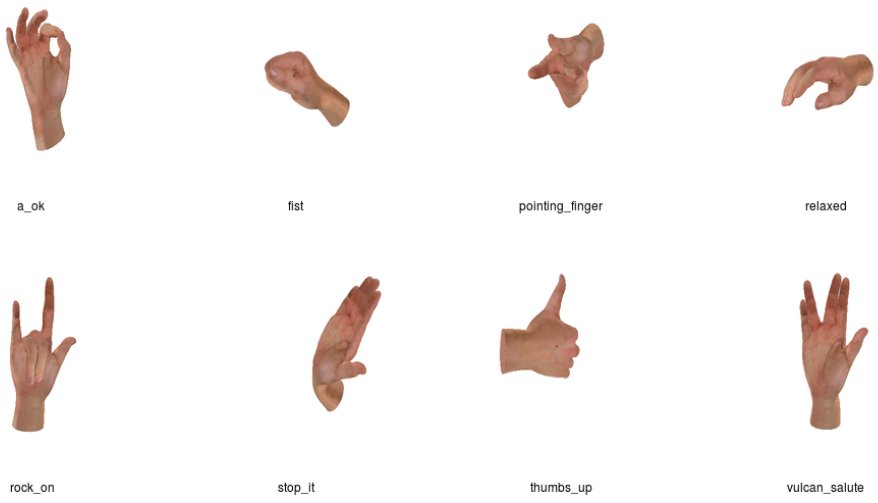


Figure 15. Default LibHand Postures

"SSIM Cube" Test System Design

To examine the feasibility image of distance as a metric for this use-case, the following testing system was designed.

A Bash script managed all of the following. A simple front end was written in Pascal. In this program, rotational and positional information was controlled via sliders. A C++ program using the Ogre3D library, which loaded the 3D model would then have the positional and rotational information piped in, coupled via Bash, and would output the rendered model in a ".PNG" file format. Another C++ program implementing SSIM iterated through the rendered image and computed the difference distances. To visualize these data an R script was employed.

To generate the output needed for testing with SSIM a software stack was developed. It was tied together using POSIX pipes and routed data such that the GUI control acted as input and the output was "SSIM cubes". This program did not use LibHand, as at the time, it was an unmaintained project and couldn't be built. However, the provided blender file of the LibHand hand was used, in addition to a low-poly custom model hand. This program, written in a combination of R (rendering and plotting), image processing (C++) Bash (scripting the pieces together) and Pascal (GUI), can be seen in Figure 16. The output of this system is shown in Figure 39 and Figure 36. To better understand the stacks, ImageJ was used as can be seen in Figure 39. This output was also processed as an image stack in ImageJ, allowing for real-time segmentation.

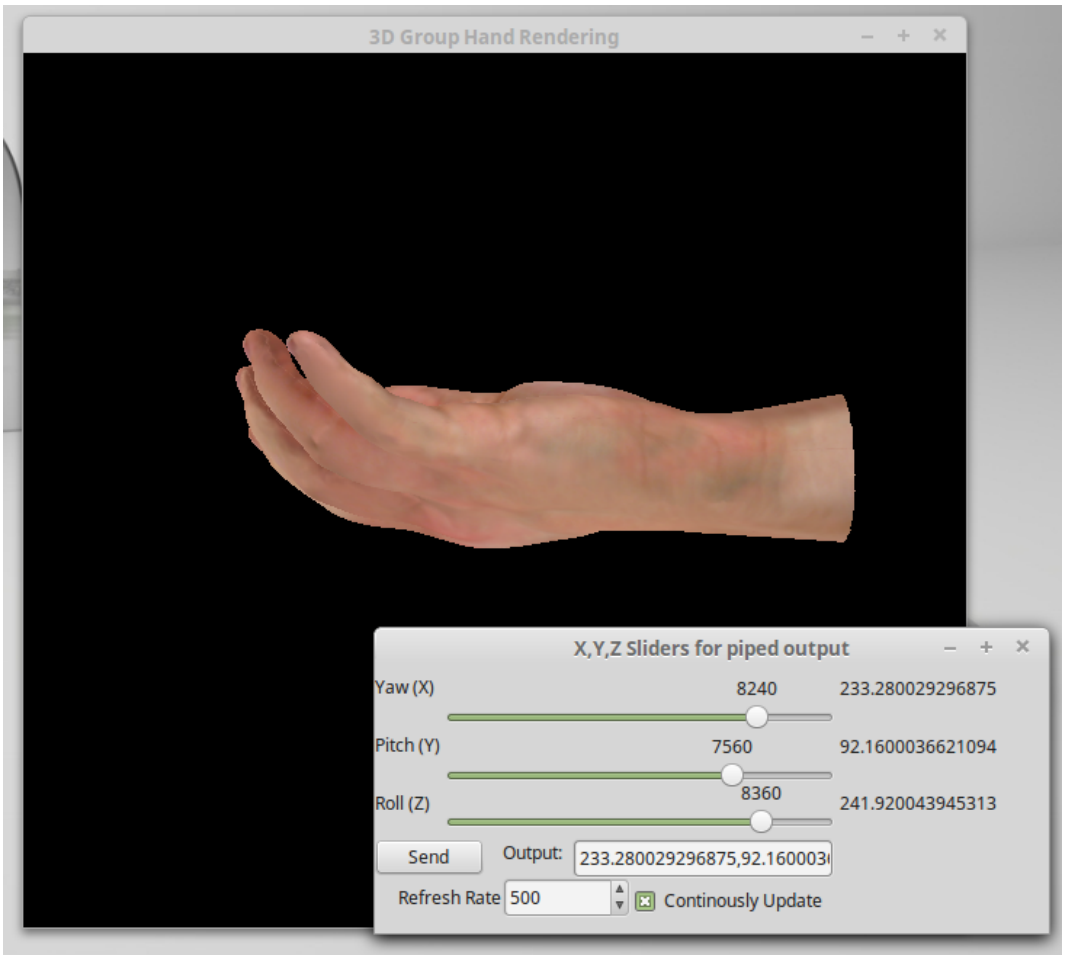


Figure 16. Manual control of 3D render rotation (fixed pose & posture).

3.2.2 Phase 2: Machine Learning System Construction

It was hypothesized that there exists at least a partial mapping between a 3D model's pose, such as one of a 3D hand, and the rendered image of such 3D model. For the set of poses for which this mapping holds true, it was considered possible to determine the inverse function by using artificial intelligence (AI).

There are three types of AI, more specifically machine learning (ML), which were deployed in this study. These biology-inspired methods are known as Artificial Neural Networks (ANN, [6, 7]), Genetic Algorithms (GA, [8, 9]), and Genetic Programming (GP, [10, 11]). The GA-generated genotypes, bitstrings, were

used to create the phenotypes, texture maps.

When this phase begun, it was unknown what was the best way to represent the GP system. For this reason, some GP functionality was manually simulated for early testing. For example, during the early testing of the Rigid_EC system, GP results were semi-randomized. This can be seen in Figure 14, where an arrow is shown under the Rigid_EC system for the GP subsystem. This was done to make testing easier and to decrease development time by isolating modules. Moreover, early planning and development had the GP system being used for both posture and pruning of the NN. This was before the custom ANN implementation was dropped in favor of using more standardized ANN libraries which provided state-of-the-art optimizations and techniques. The custom NN was implemented in a simplistic, classical Artificial Neural Network design, but the replacement NNs used were modern Deep Learning (DL) implementations. For each possible GA solution, 3D models with the appropriate textures were semi-randomly animated, which was to then be interpreted using the GP-generated decoding algorithm representing the starting point of an ANN. The ANN was then trained on the 3D model, resulting in the Reader Program (RP). Collectively, the GA, GP, and ANN software was referred to as the ML System.

The reasoning for using GA was that the number of texture maps for a given size and resolution is bounded, and can be easily represented as a linear string of bits. GP makes sense for the basis of the RP as tree structures are a natural mapping for programs where various n-ary operations can be performed and represented by branching. GP, whose variable length is its main feature, was used for encoding the ANNs. This was done to avoid human-centric biases about language and image operations from influencing the result. Future research is needed to develop a complete model of how this effects the optimization of the reader

program. ANNs are a crude imitation of how neurons cooperate to bring about complex behaviors such as those observed in the human brain. GA and GP, both types of evolutionary computing, simulate a simplified "survival of the fittest" where individual organisms are made to represent possible solutions to a problem and "evolve" over many generations, attempting to find optimums. In evolutionary computing, populations are allowed to evolve over generations approaching an optimization based on some selection of fitness. This metaheuristic search was executed in parallel for individuals of a population in a given generation. If a solution evaluated with a better score when sent through the scoring system (FE), it was assumed better and some part of the solution's encoding will have a higher likelihood of being in the next generation. Thus, while there was no study population *per se*, virtual populations were simulated over many generations as a result of the type of AI used. In this sense, population was actually referring to a dataset, and nothing physical was observed. Most interesting is that these population-based computational methods lend themselves to parallelization, enabling scalability to take advantage of computing power availability. It is noted that the effects of a multi-population simulation may be important and while such simulations were not within the scope of this work, it may be a starting place a follow-up study.

System Design

Ultimately it was decided that the renderer would be from the GP System completely. The regular application acted as a standalone and combined the incoming information from both the GP and GA systems. The resulting output was an image stack used as the input for the Neural Network system.

Several possible uses of GP were explored. From the beginning, GP was intended to be used as the scripting system for pose posture and gestures, and its use in other parts of the program were also explored. An example of how GP

could store such information is shown in Figure 17. In this tree, which acts as a descriptor of the model's armature data, bone 5's yaw is described to be rotated by π radians.

Ultimately, the use of GP was reduced to this role only. However, other possible uses included its use as pruning for the neural net system, as configuration for the reader itself, and as output from the neural net system as a simple reading program. As these roles were explored, corresponding software was developed and tested as the literature search continued. During the length of this project, advances were made leading to the lack of need for GP based pruning.

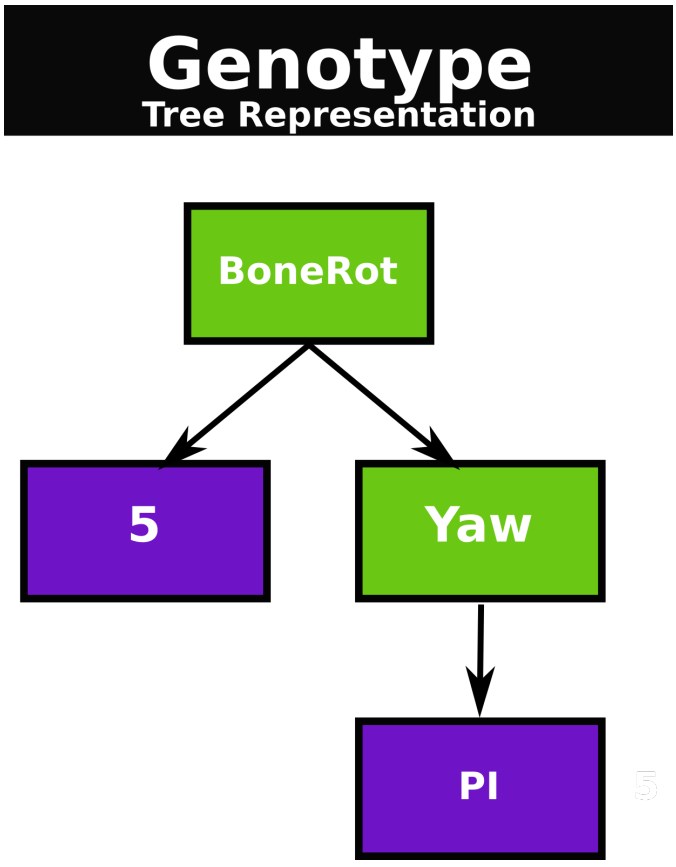


Figure 17. Example of GP storing armature information.

The rendering and GP system was written over many iterations.

1. Written in Java using the Processing library. This version loaded a 3D model

and was able to texture-map the material on it. However, while the camera and the root node of the model could be moved, the armature could not be moved.

2. Rewriting it in Python using Blender 3D's "bpy" library was also explored.

3. A LibHand clone had been started since version 0.9 could not be compiled. This used the combination of Ogre3D and Assimp libraries and was written in C++.

4. LibHand version 0.9 had serious regressions, making it difficult to use. Version 0.9z was eventually released during this study, which had fixes enabling it to be built on Ubuntu.

Several implementation iterations were explored for the neural network part of the program. The neural network system was completely written from scratch. However, this version had essentially no useful form of back-propagation, relying on pruning of the neural network as caused by the GP system. This was a state-of-the-art method until relatively recently, and matches the example implementation methods described in textbooks. Excitingly, at some point during the research, the entire landscape of the research field had a momentous change. The paradigm shift was the release as available open source software the underlining neural network systems used by Facebook, and then later Google. This was coupled with the release of faster architectures for GPUs which enabled the feasibility of modern deep learning to be included in this project. This was almost immediately incorporated, although this required significant additional learning to take place.

Evolutionary Computing Components

To test the GA system, software was written to converge toward a source image. The GP system was tested in a similar way, but instead of images as

bitstrings, polygons were rendered and were represented as linear instructions in a variant of GP, known as Linear Genetic Programming (LGP). This concept was backported to the GA system as part of the overall EC subsystem, but as a bitstring. Since the GA system went though several different versions, first written from scratch in pure C++ with OpenMP acceleration, with later versions supporting GPU, before being replaced by an industry standard tool, the Deap library for Python, this test was written for both platforms. A cartoon illustrating this process is provided in Figure 18.

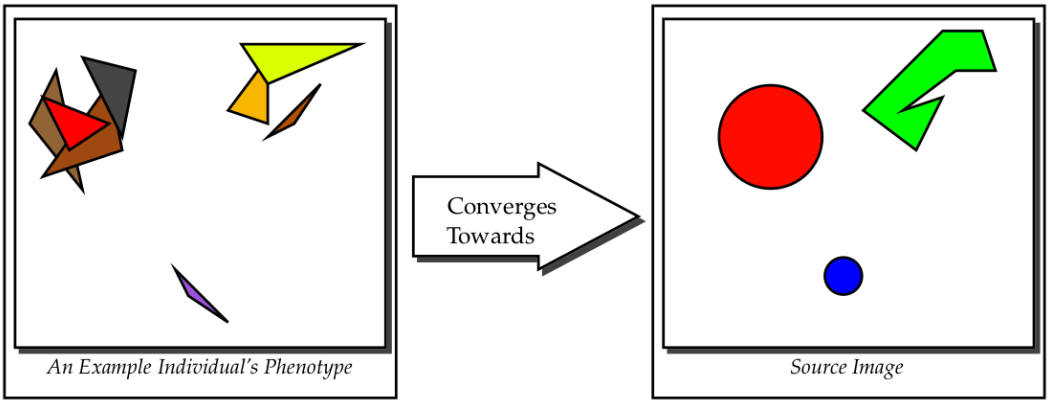


Figure 18. How a GA system might converge toward a source image.

The GP and GA systems have similar module configurations. These systems are constructed of the "Pre-Processor", the "Controller", the "Initializer", the "Evaluator", the "Fitness Calculator", the "Sorter", the "Exchanger", the "Crossover-er", the "Mutator", and the "De-Initializer" modules. This is shown in Figure 19.

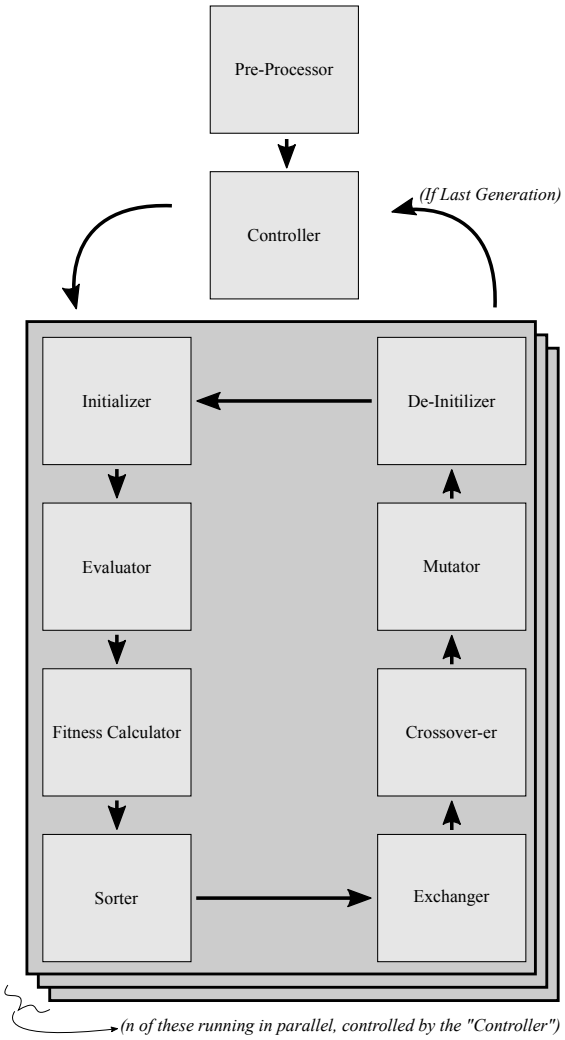


Figure 19. Overview of Flow for both GA and GP

A more in-depth explanation of the system modules is as follows:

Pre-Processor This module loads the configuration files and any previous populations to use for the initial Generations.

Controller This module handles synchronization between different civilizations. Each civilization may be running on different hardware and may not have shared memory. One of the roles of this module is to coordinate any exchanges between populations. It is also to spin up any populations which are to be executed in parallel.

Initializer Allocates memory necessary for a single generation of a given population. This module allocates memory necessary for a single generation of a given population.

Evaluator This module generates the phenotype from the genetic information for each individual.

Fitness Calculator This module coordinates with the controller to assign each individual a fitness score. This may require taking the phenotype generated by the previous step, waiting for the scores to be calculated, which may be dependent on other parts of the overall system, and then to incorporate them.

Sorter This module reorganizes the individuals based on their newly applied fitness scores.

Exchanger This module coordinates with the controller to allow for exchanges between populations existing on possibly disparate machines running in some cohort. They may not be at the same generation level, but the exchanges do require them to be in the same cohort. This exchange acts as a sort of migration.

Crossover-er This module handles crossover operations on sets of individuals.

Mutator This module adds mutations to the individuals.

De-Initializer Deallocates memory necessary for a single generation of a given population.

3D Renderer

Initial work was written in Java using the Processing (version 2 series) library for loading and importing the 3D model. There was limited skeletal control available using this library so an alternative was sought. This program was used for prototyping the initial idea. After confirming that this worked, a simple program was created using the language C++, direct OpenGL for the 3D, GLUT for the windowing control, and Assimp for model importing. While it was determined this

would be the preferable method, due to further issues with skeletal access, development of this program was paused too. The C++ library, LibHand, appeared to be a quality solution and was then explored. However, the library maintenance was in disarray, required out of date versions of Ogre3D but ultimately was usable with modifications. LibHand was removed, and instead pure Ogre3D was used instead.

To better fit into the existing literature and to take advantage of existing software libraries, it was decided to use a model of a hand that would lend itself to be compared with systems described in the literature, so that this could be applied in future work. Although it should be noted that LibHand could be extended to use other hand models, at this time only one was available. The library was extended to enable fingers to be controlled as single units and to enable dynamic texture loading. The version of the LibHand library is called in this study 0.9zj, instead of the most recent version, 0.9z, since it was a fork of 0.9 and then was merged with the fixed version of LibHand, 0.9z.

Several tests were conducted to establish how to go about the research project. These tests helped to build and determine the structure of the software engineering going forward.

3.2.3 Phase 3: Running and Evaluation

It is hypothesized that by using a variant of the host-parasite model, which I am calling the predator-prey model because my variant well matches this concept from ecology, outlined by Hillis [12], that better optimization may be achieved by avoiding stagnation of the search at local optimums. Choices for dealing with elitism, population sizes, haploid and diploid life-cycles, and other evolutionary computing strategies were evaluated as the thesis project progressed. Due to time constraints, I needed to modify the ML System's parameters, which would influence

the selection process. When adjusting the ML System's parameters occurs, it was documented. It was intended only to intervene if absolutely required because of time constraints, perturbing the ML System in the event of the solution search plateauing.

To provide additional rigor to the process of searching the solution space, I attempted to identify and use schemata, or search space subregions. This would provide a mathematical framework for evaluating solutions discovered, populations and trajectories across generations. However, this was ultimately left for future work. Mutation rate was assessed during run-time. However, as this was a solution space not previously searched, and there were be nuances unique to its surface, further research is needed for optimization. Under conditions such as a multi-generational plateau, mutation rate and other controls needed to be tweaked. However, it was decided that this optimization was outside the scope of this project and instead left for future work. Whether this would be done either by the operator or automatically is undecided at this time. Initial tests with automating it seemed feasible, however.

Connections

This project was built using a variety of platforms and applications. To connect them proved challenging. To share data, four main techniques were used:

1. Pipes (File descriptors 0/1 direct streams within system or connected through ports via netcat)
2. Messaging Queues (high-level buffered socket-like connections)
3. Files (File system storage)
4. Shared memory (Separate programs or threads which can both access each others data usually stored either on RAM or GPU)

NanoMSG is presented as a higher level socket with POSIX-like control, and was the original intended network layer. This, coupled with the inclusion of a MIT license, made it the obvious first choice. Unfortunately, NanoMSG is more recent and not complete yet across standard platform and languages, and was found to be unusable. Ultimately ZeroMQ was chosen.

3.2.4 Phase 4: Rigid Objects

Using the ML System built in Phase 2 (Subsection 3.2.2), during Phase 4 (Subsection 3.2.4) a texture map was sought that satisfied decoding requirements for rigid objects. By first simplifying the requirements and removing various possible complications such as avoiding deformations, noise, and video, it was believed the machine learning system could be guided toward an acceptable solution at a faster pace.

During this phase several instances of the ML System were run using various rigid 3D objects as inputs. In addition to a rigid 3D model of a human hand, 3D primitives such as a cube, cylinder, sphere, and the classical Utah Teapot were also used [13, 14]. The deliverable at the conclusion of this phase, was a texture-mapped rigid 3D model of a human hand and complementary reader which together are valid for a range of poses within a limited tolerance.

It is hypothesized that topologically related objects may be related through their ML System solutions, working texture maps and readers, such that a solution for one 3D object is found to be a reasonably good starting point for solving another.

Rigid Objects: Cube

Figure 20 shows a cube rendered with three different sized encoding dimensions. The texture maps are 16×16, 64×64, and 256×256. Another cube texture

map representative of a 6 sided die, something which is relatively easy for humans to read the 3D orientation of, was also tested and is shown in Figure 21.

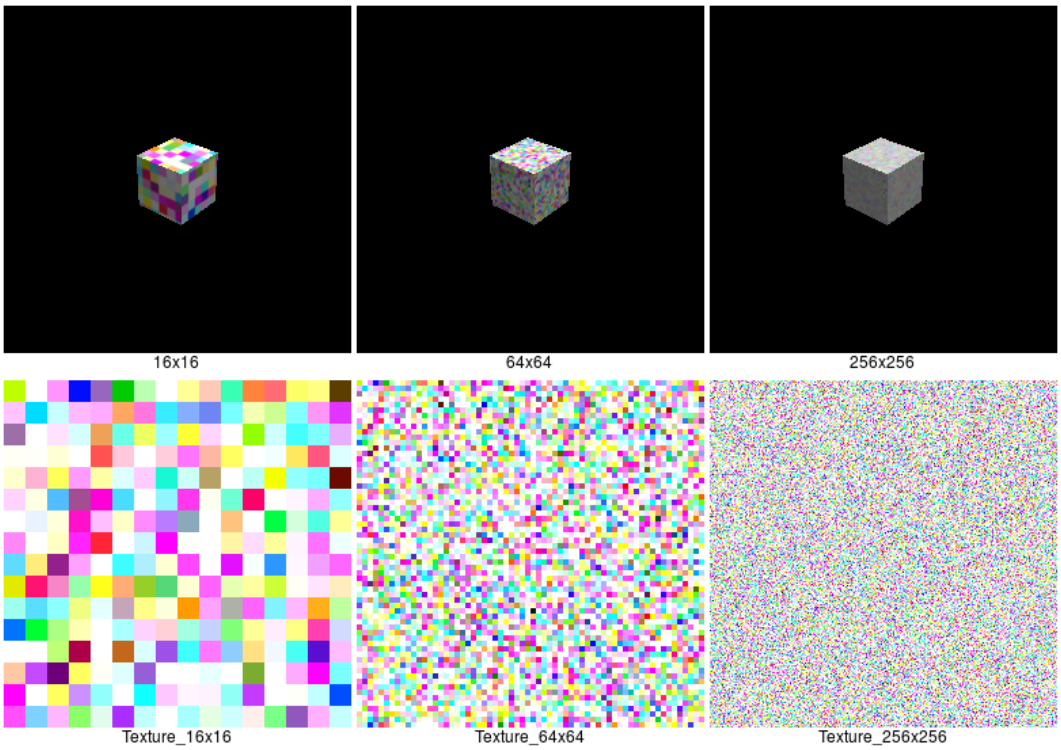


Figure 20. Cube textures

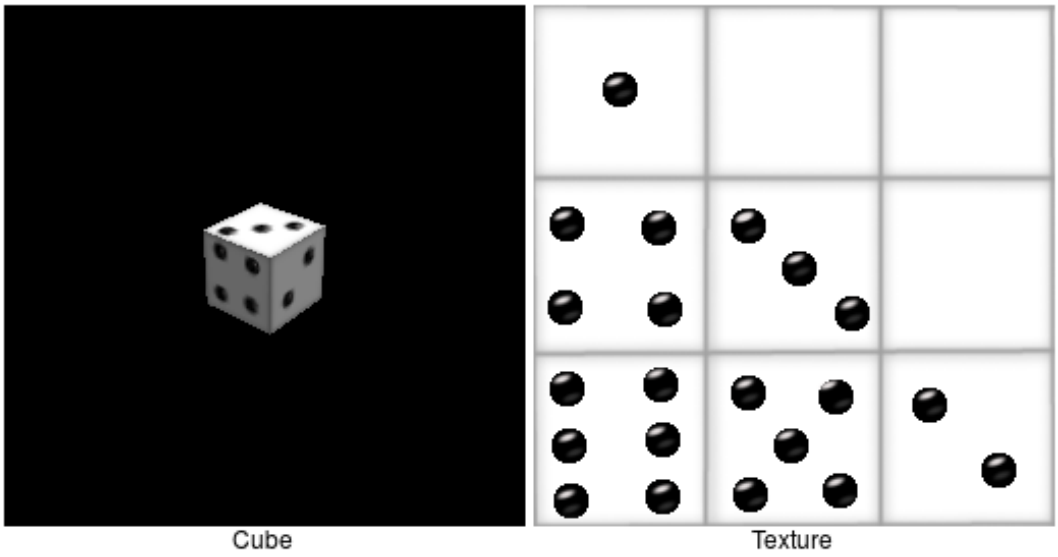


Figure 21. Dice Cube textures

Hand texture

The texture map at this resolution was determined to be too much for the available testing systems, but provided better hardware, a follow-up study should take advantage of the higher resolution texture map. To determine the lowest resolution texture map which ideally would not require a significant trade-off of usable available encodable data available to the system, several texture map input and output render sizes were compared.

Figure 22. Default LibHand texture map.

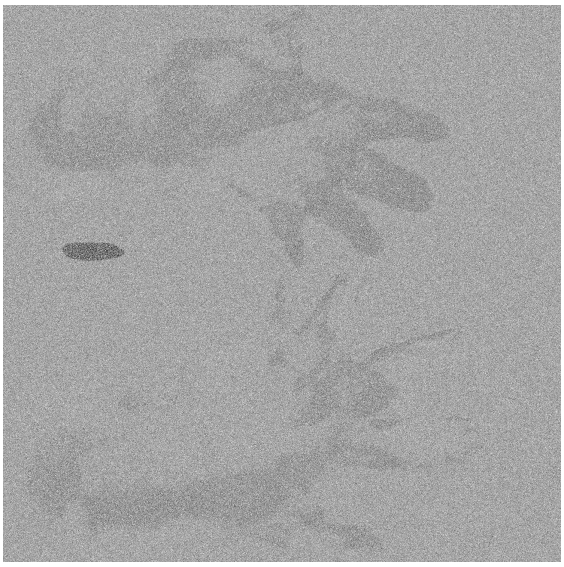


Figure 23. Texture map for testing encoding capabilities. There is a semi-transparent layer of a white background with RGB noise, on top of the default texture map of LibHand.

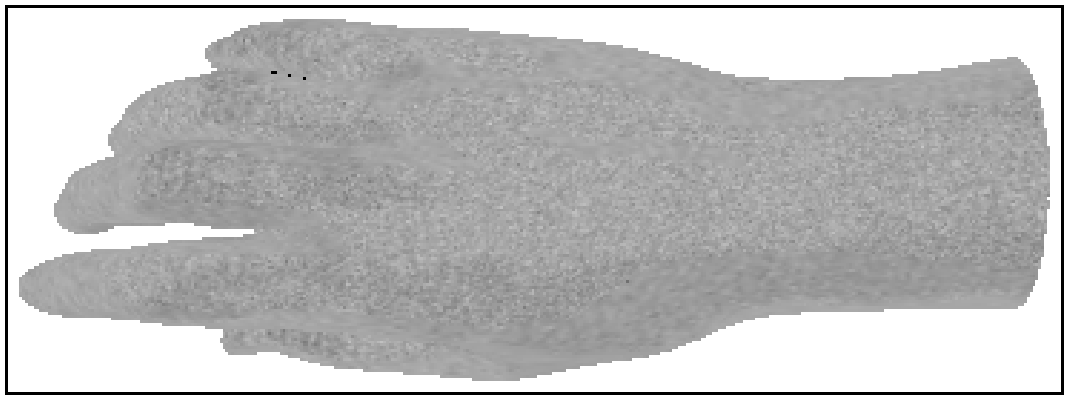


Figure 24. Render of relaxed hand using a the texture map from Figure 23.

To test the system and, moreover, to test the presumptions about the system, the following experimentation needed to be executed.

Figure 24 shows this texture map rendered onto a natural relaxed posed version of the hand.

Figure 22 shows the source texture map.

The modified version is at Figure 23. This demonstrates a rendering on a

white source backgrounds with red, green, and blue noise randomly generated. This can be compared with the original texture map provided by LibHand.

All texture maps explored during this thesis are shown in Figure 25. Figure 26 shows the texture maps used for generating the rendered posture datasets.

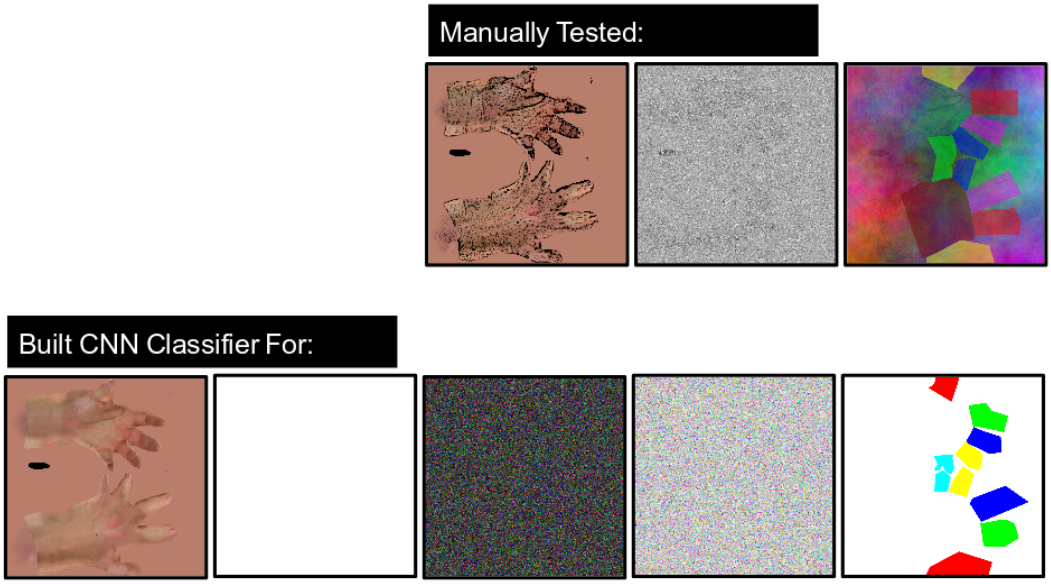


Figure 25. Texture maps explored in thesis.

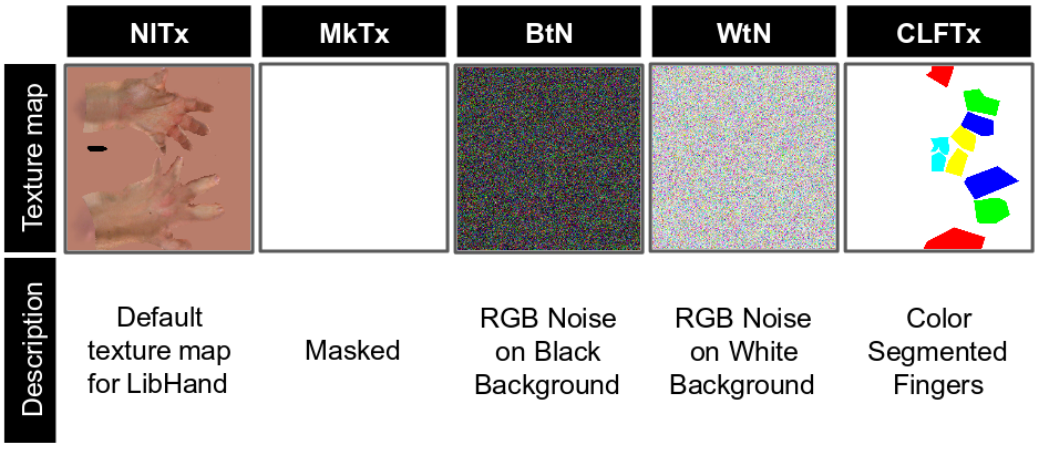


Figure 26. Texture maps used for generating the rendered posture datasets.

Additional Pose Categories

To further test the system, new hand signs for “B”, “C”, “D”, “M”, “Y”, and “flat” were produced. Figure 27 and Figure 28 show “C” from the LibHand Pose Designer. These new poses are shown in Figure 29. Finally, the complete listing of the 14 postures used in this study can be found in Figure 30. To make it easier to run experiments, abbreviations were given to all 14 postures, as shown.

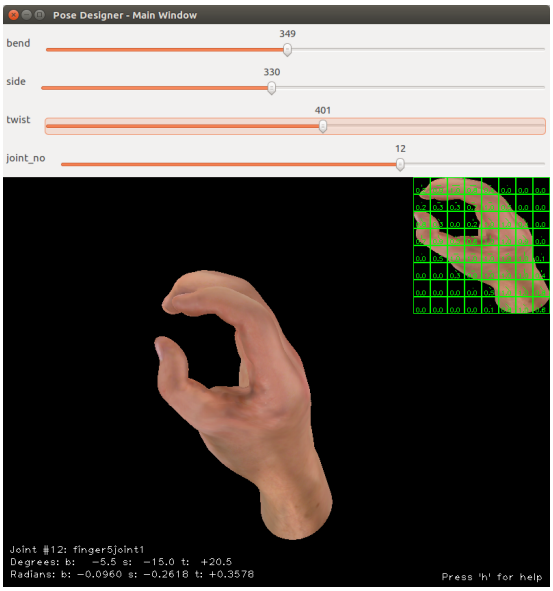


Figure 27. Creating pose “C”. Side view.

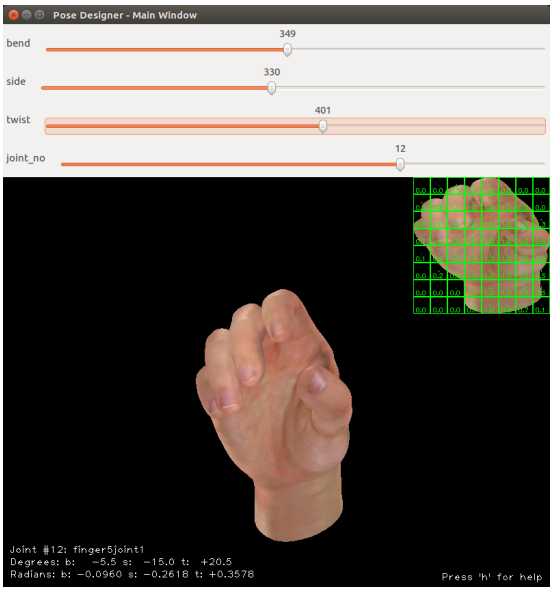


Figure 28. Creating pose "C". Front view.

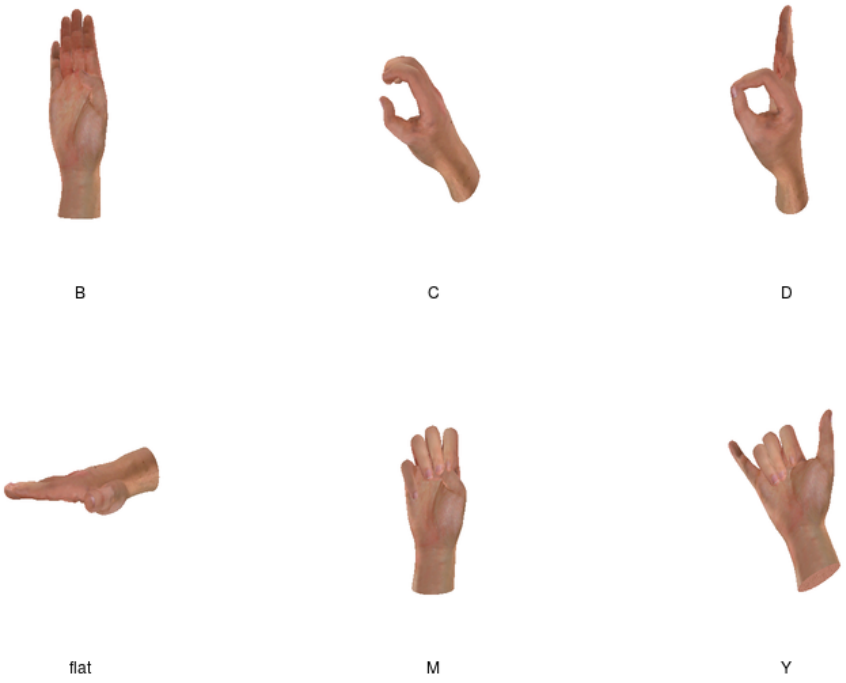


Figure 29. New LibHand Postures

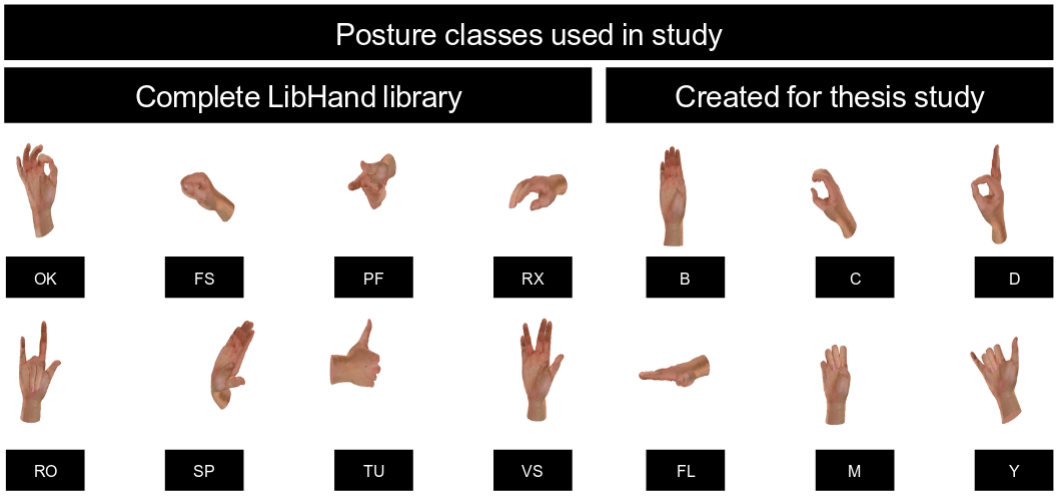


Figure 30. Complete listing of the 14 postures used in this study and their abbreviations.

Masks and Experimenting with Pose Classification

Going into the project it really wasn't clear whether the texture maps would provide enough information to determine pose or posture. Furthermore it needed to be determined how much the perimeter of the object played a role in the neural net being able to identify the pose. For this reason a series of exploratory experiments were performed. The assumption was that insight could be gained by producing masks of the object, removing everything but the perimeter by segmenting it out of the image, and using this to train the neural network classifier.

3.2.5 Phase 5: Deformable Objects

Finally, deformable objects were tackled. In this phase an articulated 3D model of the human hand was studied. This let to a new solution to the glove-based human hand gesture recognition problem of computer vision and was evaluated. The rigid 3D model of the hand that was previously used was now allowed to deform as a real hand might. Whether the deformation method used would be based on the many lattice or armature methods was undecided. However, the latter was thought to be more likely. The physiology of the human hand is well documented

and many models already exist [15, 16, 17, 18]. The specific representation that will be used for the simulation is not yet determined and the decision will require experimenting with available options. Any model selected will likely have between 6- to 26- degrees of freedom, although only 20 may be needed [19, 20, 21]. However, these may not all be accessible due to implementation of inverse kinematic chains (or approximation thereof).

Taxonomies and notations methods such as Choi *et al.*'s work on hands, Müller *et al.*'s work on general motion capture may be used to represent or describe the 3D model's postures [22, 23]. It is hypothesized that by using the likelihood of a specific posture based on gesture comfort as an additional component to a solutions fitness may lead to better results and may be explored in the future [22].

Redesign of the Dimensions of the Problem

Initially the plan was to use a model based on previous work by F. Lathuilière *et al.* [24]. However, it was decided that the degrees of freedom offered by this model were too high even though it would be possible to use and preferable due to it being a more realistic model. Instead, the 3D hand model and rig was recreated with a reduced dimensionality in hope of reducing the complexity and thus time required for training.

An analysis of how the required training set's size scales with the model complexity is presented in the analysis section (Chapter 4).

Resolution of the Training Set

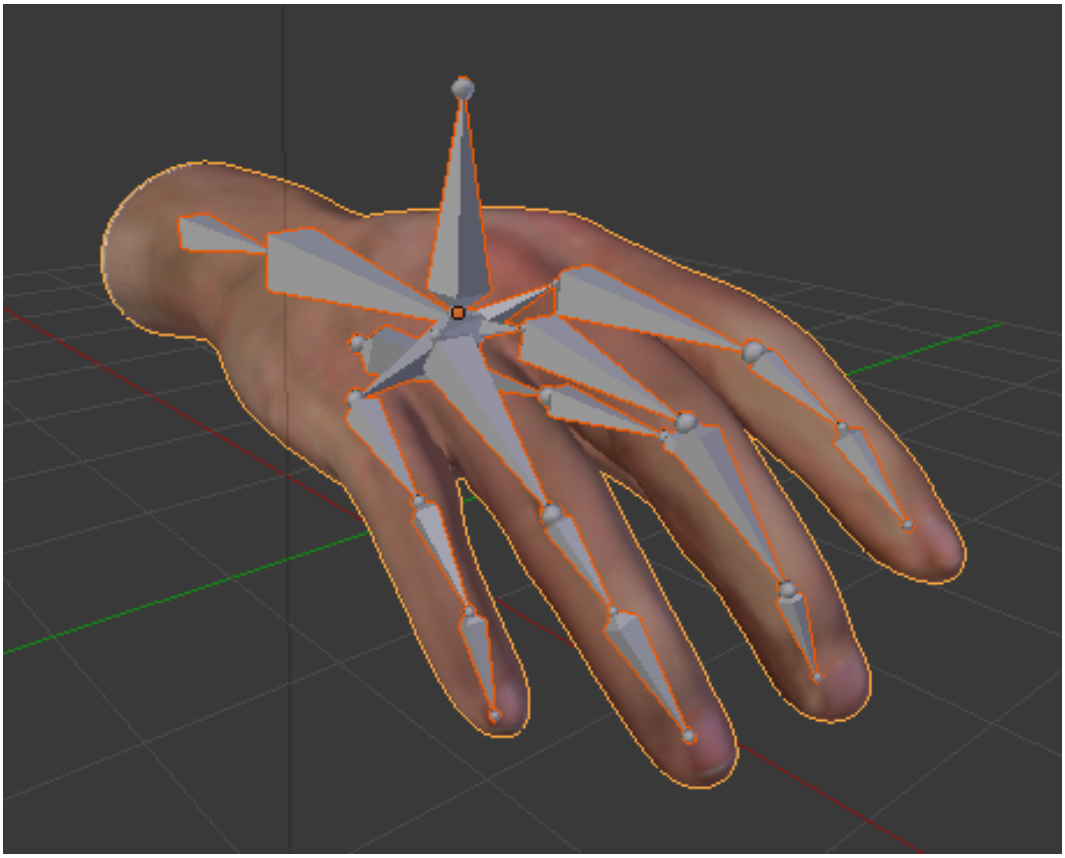


Figure 31. This image shows the skeletal armature control of the default LibHand model from a 3/4ths view, as rendered in Blender 3D. Note the obscured thumb bone structure.

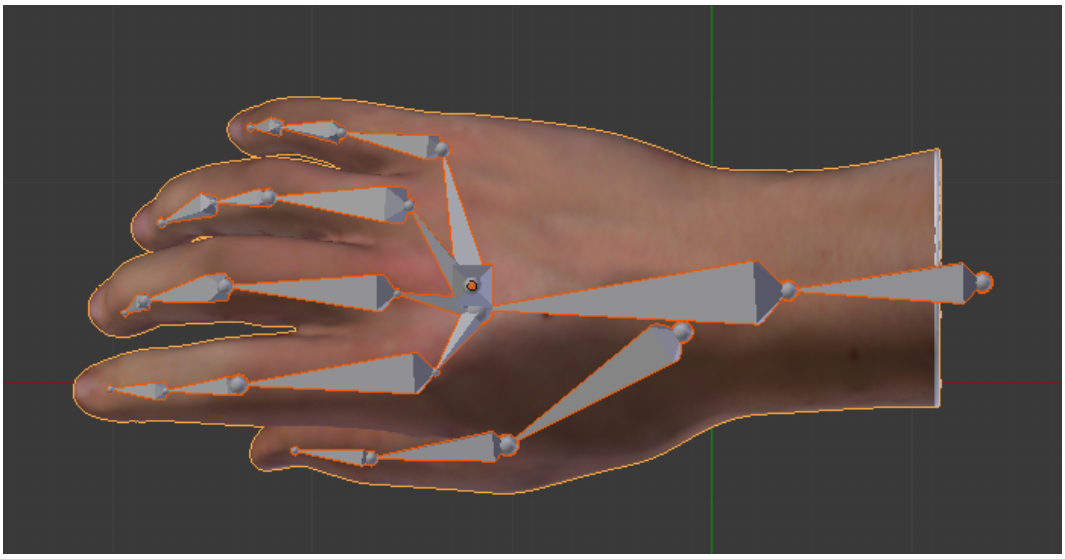


Figure 32. This image shows the skeletal armature control of the default LibHand model from a top-down view, as rendered in Blender 3D. Note the obscured palm bone structure.

Before incorporating posture capabilities into the system, an assessment of the hand structure that LibHand uses, was performed. The default model was a high-polygonal mesh with a large texture map. A Blender3D source model file and converted, usable but outdated version for Ogre3D were both provided as part of the LibHand source repository. This is shown in Figure 31 and Figure 32. The armature or bone structure used in LibHand can be seen here.

To have the system run in a reasonable amount of time, the model had to be optimized to a lower polygon count version, with precomputed vertexes for Ogre3D, and a lower resolution texture map. Furthermore, the DoF available within LibHand was too large for the available computing equipment, and thus the range of control needed to also be reduced. However, it required some evaluation to determine the optimal reduction for these.

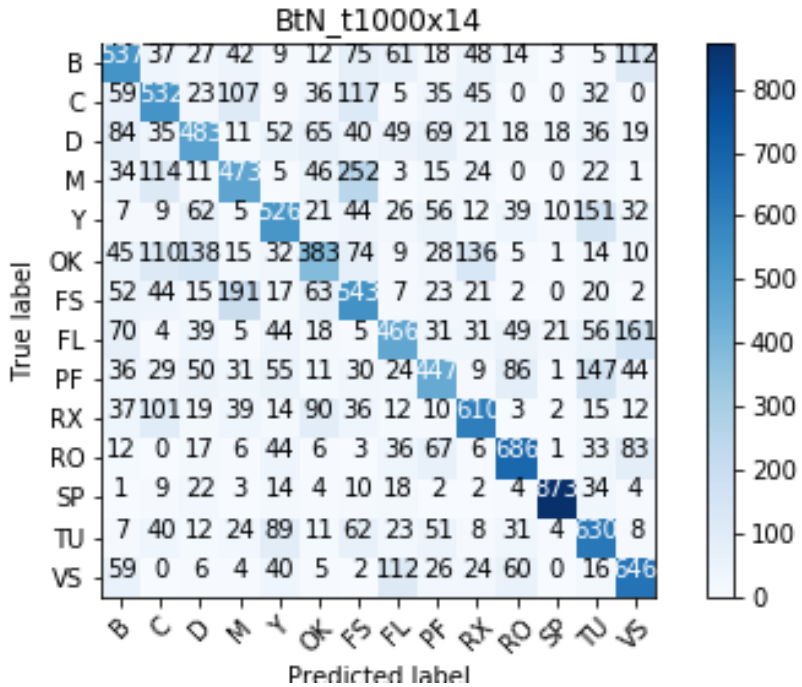


Figure 33. Experiment BtN_t1000x14: Confusion matrix where system used the BtN texture map, 14 postures, and 1000 images per class.

3.2.6 Phase 6: Noise

Special considerations have been given to dealing with noise. For example, the image noise, environmental effects, and image background must all be considered and accounted for. A better texture map and reader solution had to have some tolerance for noise issues. Such imperfections initially needed to be simulated, and it was hypothesized that doing so would help to build-in resistance to issues related to those simulated as they will inevitably show up during a glove's use.

Although it was thought in the beginning that rendered images might be segmented, this was not implemented. During the initial training phase, the image background was be a flat color, however during later stages, the ability to use a more noisy background such as photographs or film was implemented. However, this was not explored and is left for future work. For video frame differencing, background subtraction should likely also be used. While very different in methods

and objectives overall, work done by Wang *et al.* on identifying hand shape in an image provided an excellent starting place for hand shape extraction methods, as they also used GA [25].

Time concerns were not addressed due to the limitations and time constraints of a Masters thesis.

List of References

[1] A. Bovik *et al.*, *Handbook of Image & Video Processing*, 2nd ed., A. Bovik, Ed. Burlington, MA, USA: Elsevier Academic Press, 2005.

[2] R. Dosselmann and X. D. Yang, "A comprehensive assessment of the structural similarity index," *Signal, Image and Video Processing*, vol. 5, no. 1, pp. 81–91, 2009. Accessed on April 15, 2018. Available: http://dx.doi.org/10.1007/s11760-009-0144-1

[3] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, vol. 2, Pacific Grove, CA, USA, Nov. 2003, pp. 1398–1402 Vol.2.

[4] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[5] D. H. Ballard and C. M. Brown, *Computer Vision*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc, 1982.

[6] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed., ser. MIT Press. Cambridge, MA, USA: A Bradford Book, 1995.

[7] K.-L. Du and M. N. S. Swamy, *Neural Networks and Statistical Learning*, 1st ed. NY, USA: Springer, 2013.

[8] M. Mitchell, *An Introduction to Genetic Algorithms*, 1st ed. Cambridge, MA, USA: The MIT Press, 1996.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 1992.

[10] J. R. Koza, F. H. Bennett, III, D. Andre, and M. A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., 1999.

[11] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming An Introduction On the Automatic Evolution of Computer Programs and Its Applications*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann, 1998.

[12] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, pp. 228–234, 1990.

[13] M. E. Newell, "The utilization of procedure models in digital image synthesis," Ph.D. dissertation, University Of Utah, UT, USA, January 1975, original source of the Utah Teapot also know as the Newell Teapot.

[14] A. Torrence, "Martin Newell's original teapot," in *Proceedings of the ACM SIGGRAPH*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. Accessed on April 15, 2018. Available: http://doi.acm.org/10.1145/1180098.1180128

[15] I. Albrecht, J. Haber, and H.-P. Seidel, "Construction and animation of anatomically based human hand models," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 98–109. Accessed on April 15, 2018. Available: http://dl.acm.org/citation.cfm?id=846276.846290

[16] T. Rhee, U. Neumann, and J. P. Lewis, "Human hand modeling from surface anatomy," in *Proceedings of the Symposium on Interactive 3D Graphics and Games*, ser. I3D '06. New York, NY, USA: ACM, 2006, pp. 27–34. Accessed on April 15, 2018. Available: http://doi.acm.org/10.1145/1111411.1111417

[17] M. Šarić, "Libhand: A library for hand articulation," 2011. Accessed on April 15, 2018. Available: http://www.libhand.org/

[18] E. Shasheen, M. Šarić, *et al.*, "Libhand: A library for hand articulation," 2014, version 0.9.z. Accessed on April 15, 2018. Available: http://www.libhand.org/

[19] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, 2007, special Issue on Vision for Human-Computer Interaction. Accessed on April 15, 2018. Available: http://www.sciencedirect.com/science/article/pii/S1077314206002281

[20] J. Lin, Y. Wu, and T. S. Huang, "Modeling the constraints of human hand motion," in *Proceedings of the IEEE Workshop on Human Motion*, Austin, TX, USA, 2000, pp. 121–126.

[21] Y. Wu and T. S. Huang, "Hand modeling, analysis and recognition," *IEEE Signal Processing Magazine*, vol. 18, no. 3, pp. 51–60, May 2001.

[22] D. Rempel, M. J. Camilleri, and D. L. Lee, "The design of hand gestures for human-computer interaction: Lessons from sign language interpreters," *International Journal of Human-Computer Studies*, vol. 72, no. 10–11, pp. 728–735, 2014. Accessed on April 15, 2018. Available: http://www.sciencedirect.com/science/article/pii/S1071581914000706

[23] E. Choi, H. Kim, and M. K. Chung, "A taxonomy and notation method for three-dimensional hand gestures," *International Journal of Industrial Ergonomics*, vol. 44, no. 1, pp. 171–188, 2014. Accessed on April 15, 2018. Available: http://www.sciencedirect.com/science/article/pii/S0169814113001285

[24] F. Lathuilière and J. Y. Hervé, "Visual tracking of hand posture with occlusion handling," in *Proceedings of the International Conference on Pattern Recognition*, vol. 3, Barcelona, Spain, 2000, pp. 1129–1133 vol.3.

[25] J.-W. Wang, C.-C. Wang, and J.-S. Lee, "Genetic eigenhand selection for handshape classification based on compact hand extraction," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2215–2226, 2013. Accessed on April 15, 2018. Available: http://www.sciencedirect.com/science/article/pii/S0952197613001218

CHAPTER 4

Experimental Analysis

In this chapter, Chapter 4, the results of the conducted experiments are presented, and an analysis of their potential meaning is interpreted. As in Chapter 3, the chapter is arranged as a set of project phases. These are found in Subsections 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.1.5, and 4.1.6.

4.1 Phases

4.1.1 Phase 1: Numerical Measurement Methods

Presented here are the results of experiments described in Phase 1 (Subsection 3.2.1).

Significant progress in computer vision has been made since the original proposal was submitted. Because scientific progress was made since the initial proposal, the experiment needed to be adjusted. To take advantage of these changes and place this project within the context of this dynamic progress, it is necessary to compare.

It is now possible to build a fairly robust classification software which can identify poses of a hand without a glove. However, the core question to this research has always been concerned with the aid that additional encoding may provide.

The encoding step ran for an extended period of time to prepare the system. This generated a texture with sufficient differences at close postures to be meaningful. The Neural Network side was then engaged.

This all was then compared against the current standard. Two machines were used, one running one system and the other running the other. Had there been more time for the tasks, each machine would then flip. That would be done as a

control.

Interpretation of the "SSIM Cubes"

Figure 34 shows the front view, side view, and the frames of repeating circular animation of the Yaw SSIM stacks rotating on Yaw, having been rendered in ImageJ to show clusters of rotations with similar SSIM values. The clusters are likely caused by perspective foreshortening, making angles where the hand is facing the camera appear to change more rapidly per rotation step.

Examples of a sliced SSIM cube, where the red, green, and blue channels represent the SSIM values between rotations in yaw, pitch, and roll, respectively, as intensity, are presented in Figure 35. Additional SSIM cubes can be seen in Figures 36 and 37. Figure 38 shows the data binned for easier interpretation.

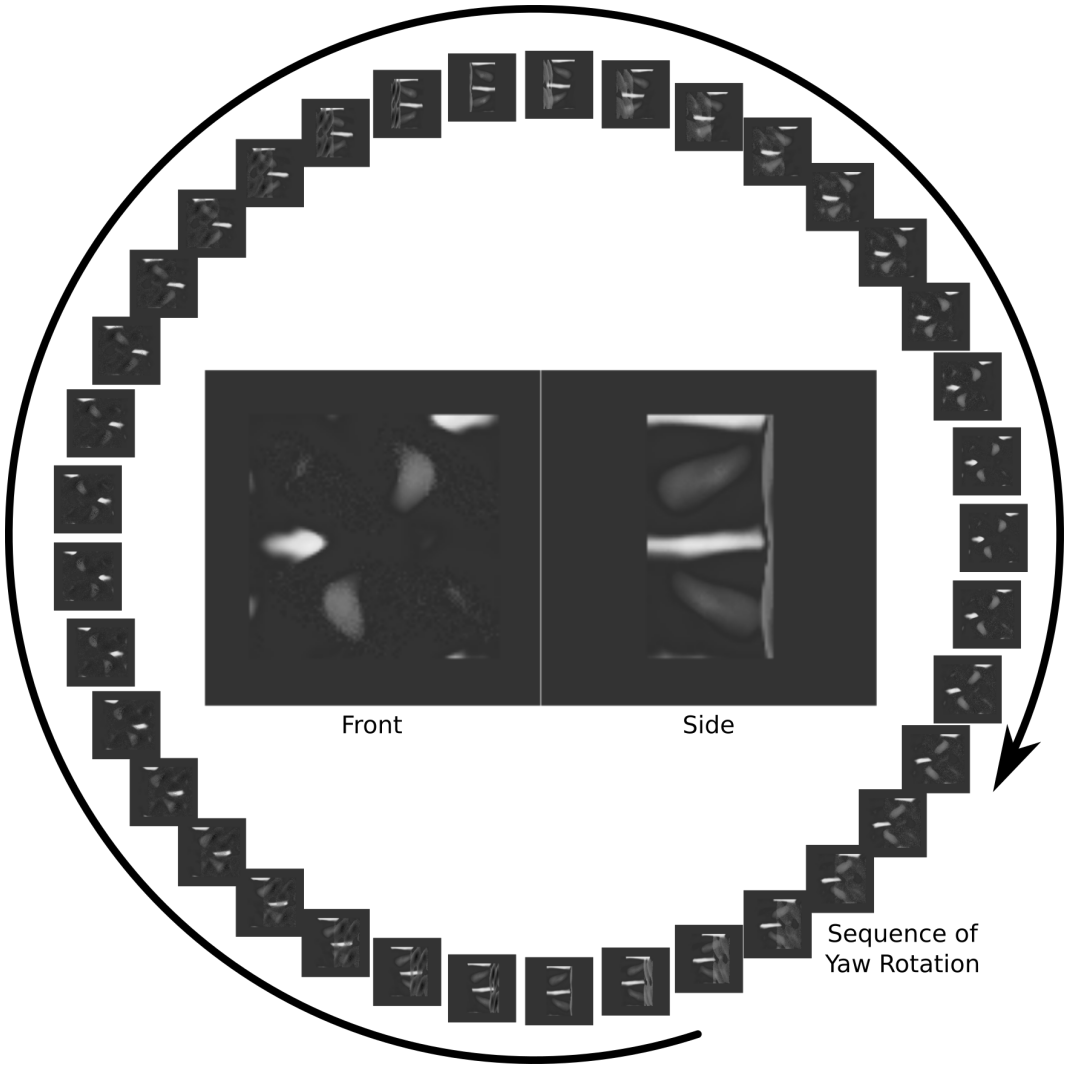


Figure 34. SSIM Testing.

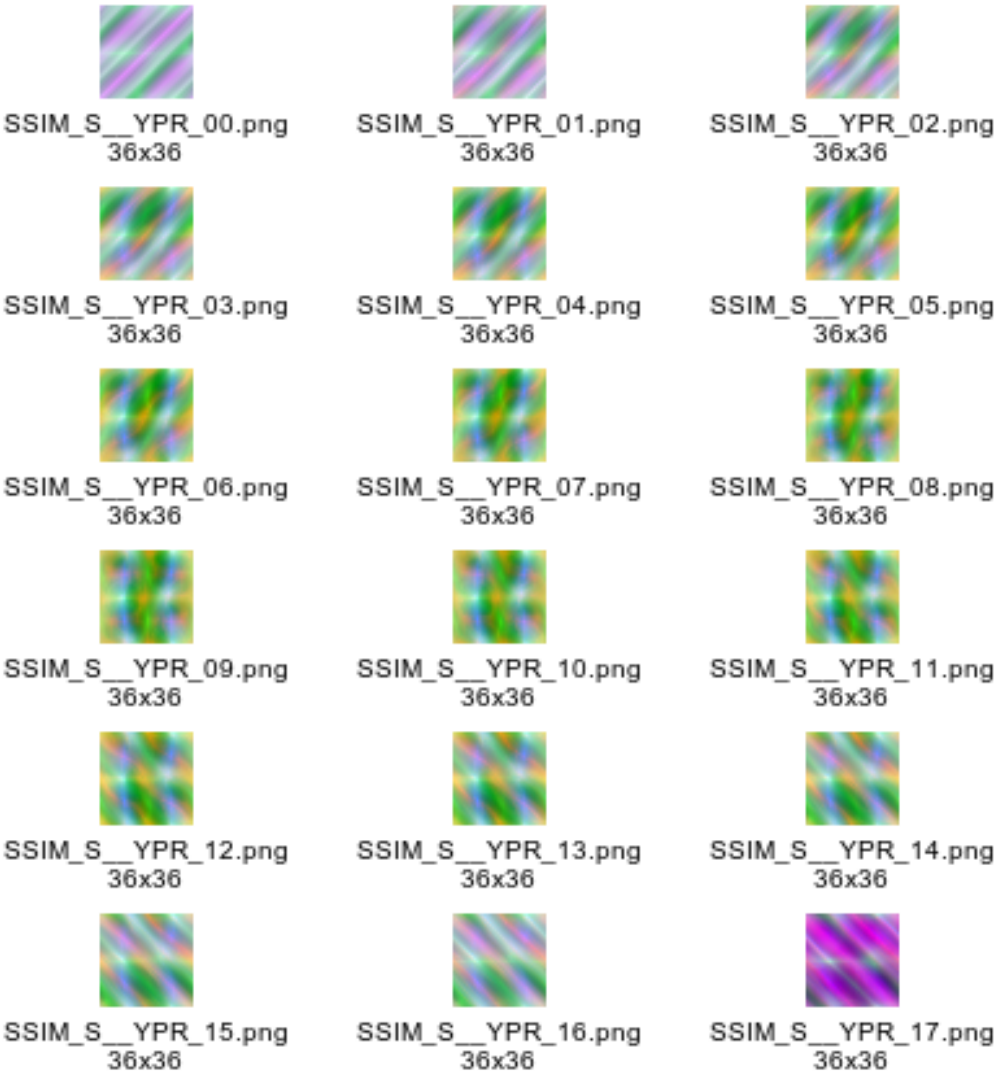


Figure 35. SSIM S YPR montage

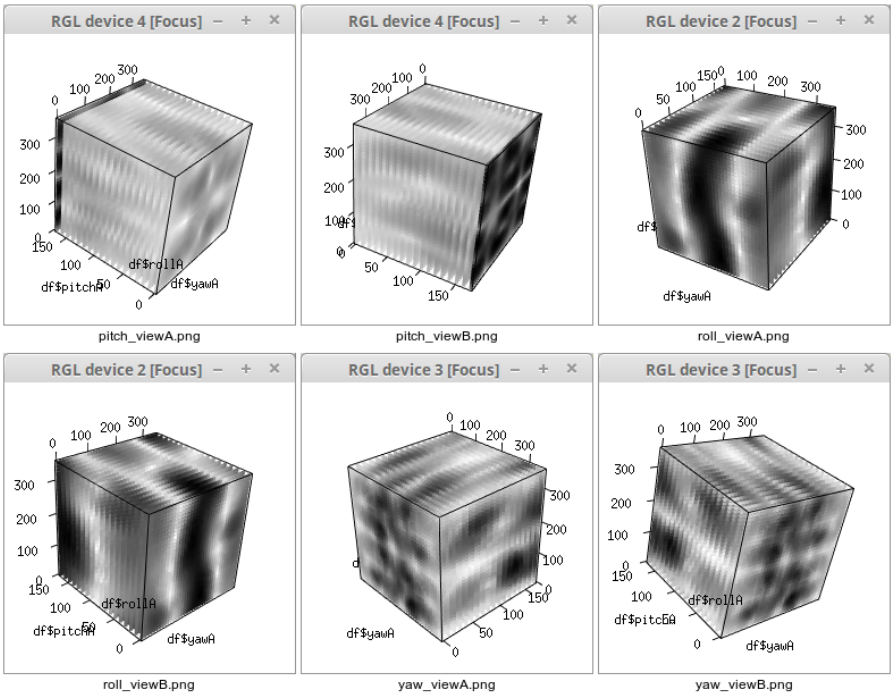


Figure 36. Volumes from SSIM image stacks

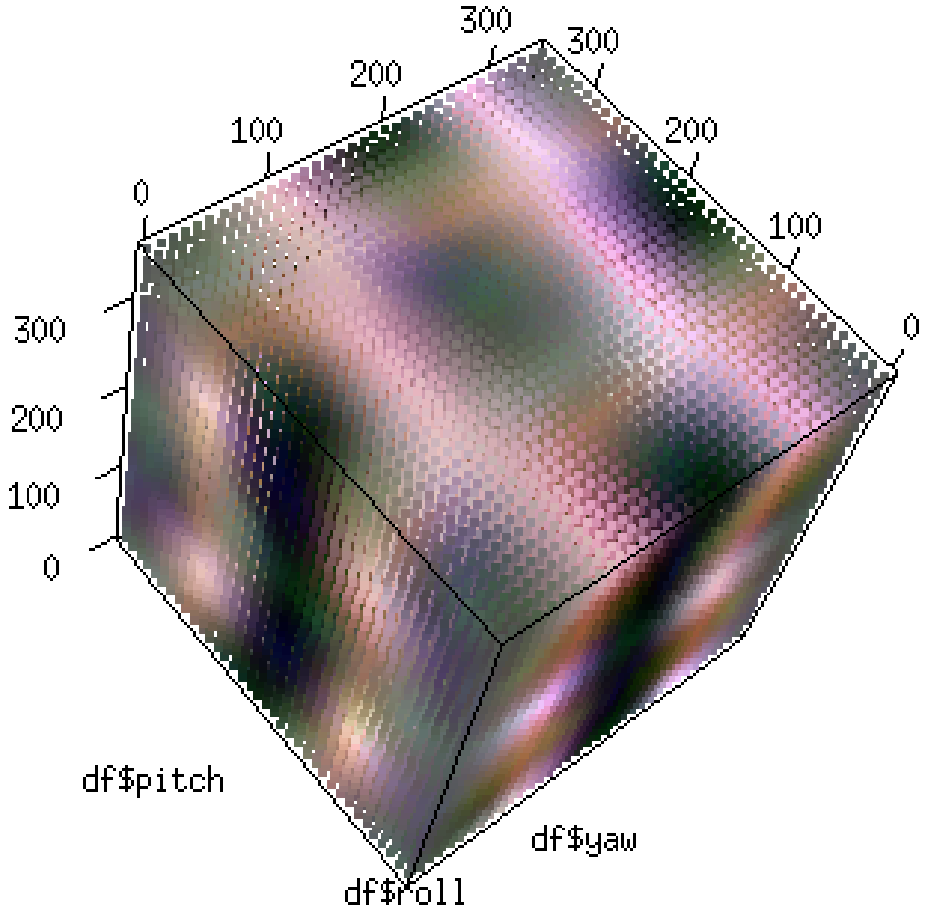


Figure 37. Volumes from SSIM image stacks

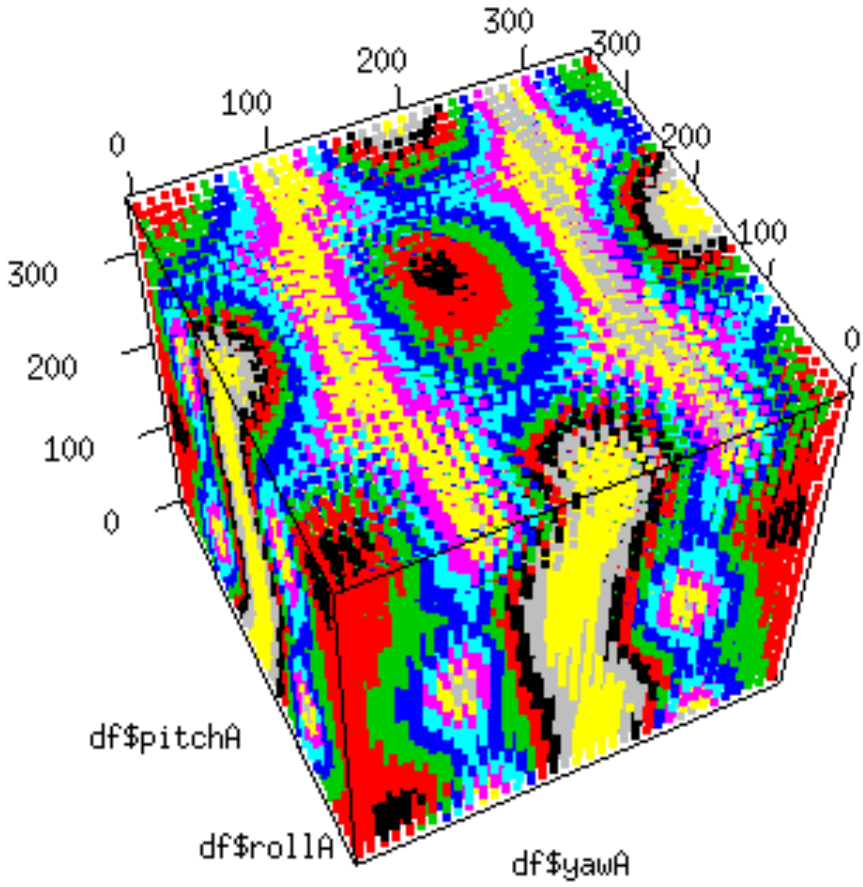


Figure 38. SSIM cube with colorized data binning

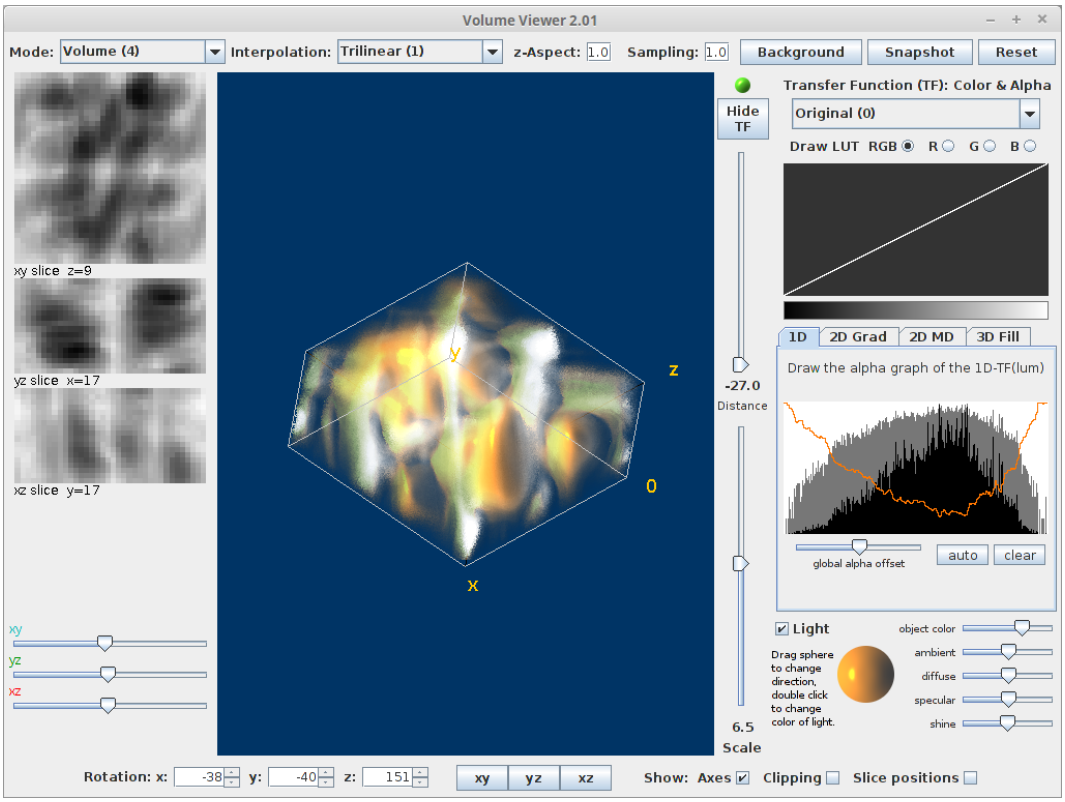


Figure 39. Volumetic rendering of of SSIM scores between close rotations of +/-5 degrees (fixed pose & posture). *Screenshot of ImageJ with Volumetic Image Stack plugin.*

4.1.2 Phase 2: Machine Learning System Construction

Presented here are the results of experiments described in Phase 2 (Subsection 3.2.2).

After writing the GA system, it was necessary to test. To evaluate whether or not the GA system would be able to assist in directing the overall system toward convergence on an ideal texture map, the GA system was tested by itself. It was made to converge toward a known texture map over 9950 generations. This is shown in Figures 40 and 41. In the first image, opaque polygons were represented by the bitstrings denoting the vertex locations and color. This was done primarily on the CPU. In the second image, a more GPU-intensive version of the same process was performed where the difference was transparency on the

triangle polygons. Both of these versions were written in C++ with a custom GA. Later on in the project, after moving primarily to Python, in order to make the system more comparable to other works, the Python library Deap was employed, as discussed previously. Since it was already established that a GA system could converge toward a source image using complex shapes, this part of the experiment was only checking whether or not the pixel was the same, rather than the distance in color. Although this was not a rapid process, it did isolate some of the nuances of writing the other parts of the system.

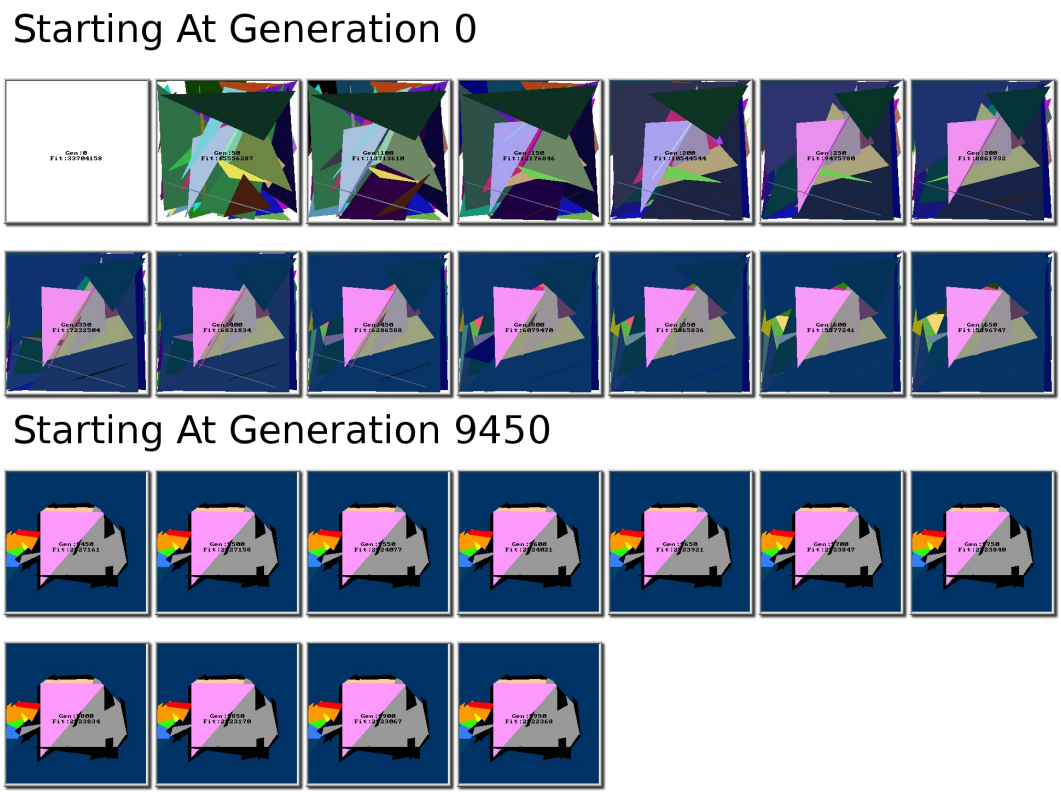


Figure 40. Testing C++ GA system to converge on a source image using opaque shapes (Original Version)

Figure 41. Testing C++ GA system to converge on a source image using translucent shapes (Original Version)

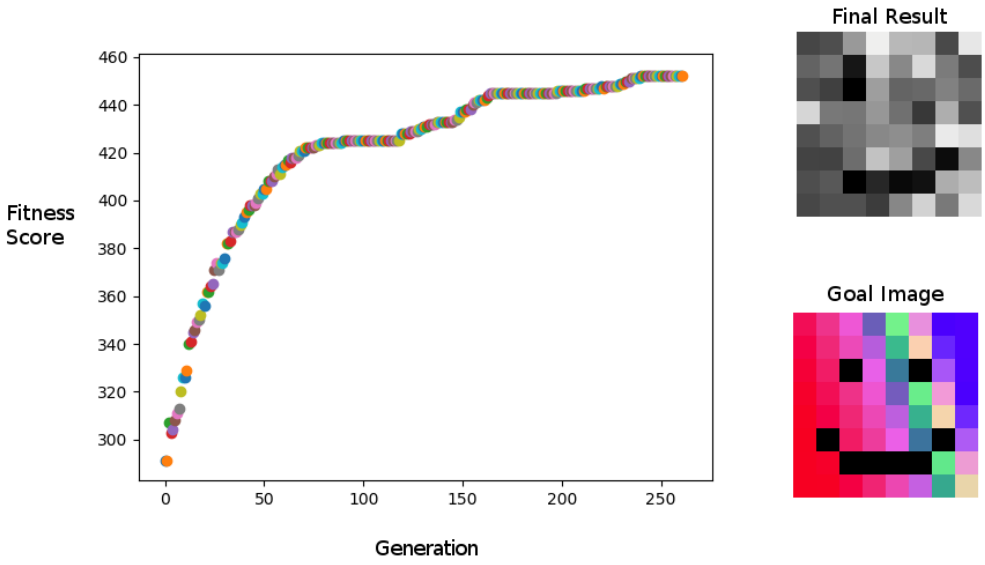


Figure 42. Testing Python GA system to converge (Deap Version)

Figure 42 shows an example test run of the Deap system implementation, converging 256 generations toward the source image shown as the goal. As is common with EC systems, there was a rapid improvement phase, followed by a rapid tapering off of the progress in a way that appeared asymptotic.

4.1.3 Phase 3: Running and Evaluation

Presented here are the results of experiments described in Phase 3 (Subsection 3.2.3).

The 100 by 100 pixel image pose, single axis rotation test is shown in Figure 43. After including the LibHand model, a similar test was done, as shown in Figure 44.

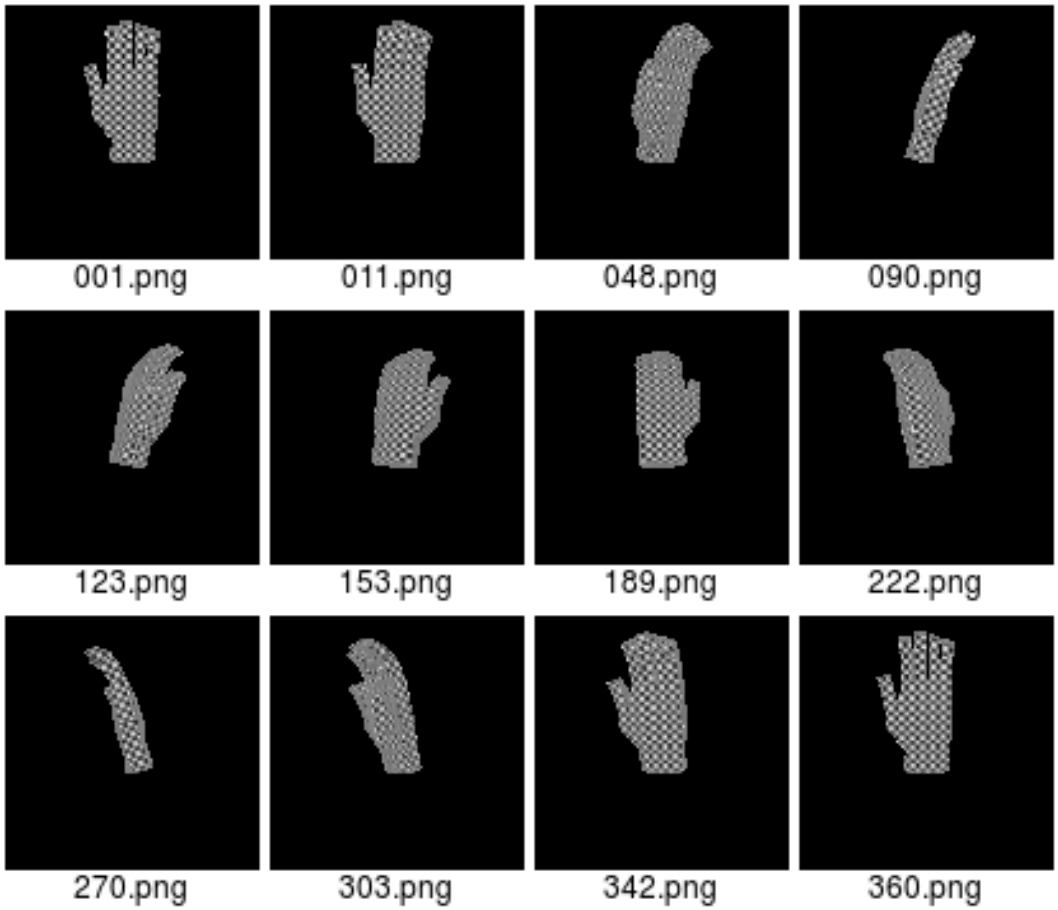


Figure 43. Rotating 100 ×100 small sample

Figure 44. Rotating hand with noise texture map showing "2D Barcode" example. From left to right renders with roll of 100, 110, and 120, all with yaws and pitches of 100.

It was observed during testing that some of the postures were easier to classify for the system than others. Upon inspection, it was found that the choice for not rendering any data on the cut-off part of the wrist and extended fingers were

influencing the system due to their contours. Figure 45 shows the use of GradCAM in assessing the influence of input pixels on the resulting classification from the NN system [1].

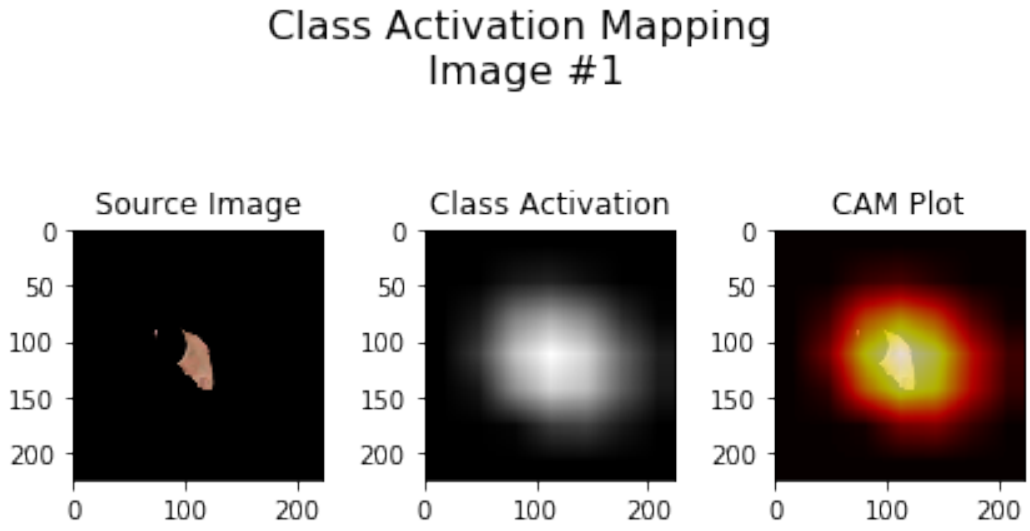


Figure 45. Looking at which input pixels influence the indicated classification the greatest

To get a better idea of the influence, masks were created of the postures by thresholding all non-background pixels in the NlTx dataset and maxing their red, green, and blue channels. Figures 46 and 47 show good examples of both convex and deflection caused by the fingers moving away from the center of mass, and the non-rendered wrist, both resulting in arguably identifiable contours of postures.

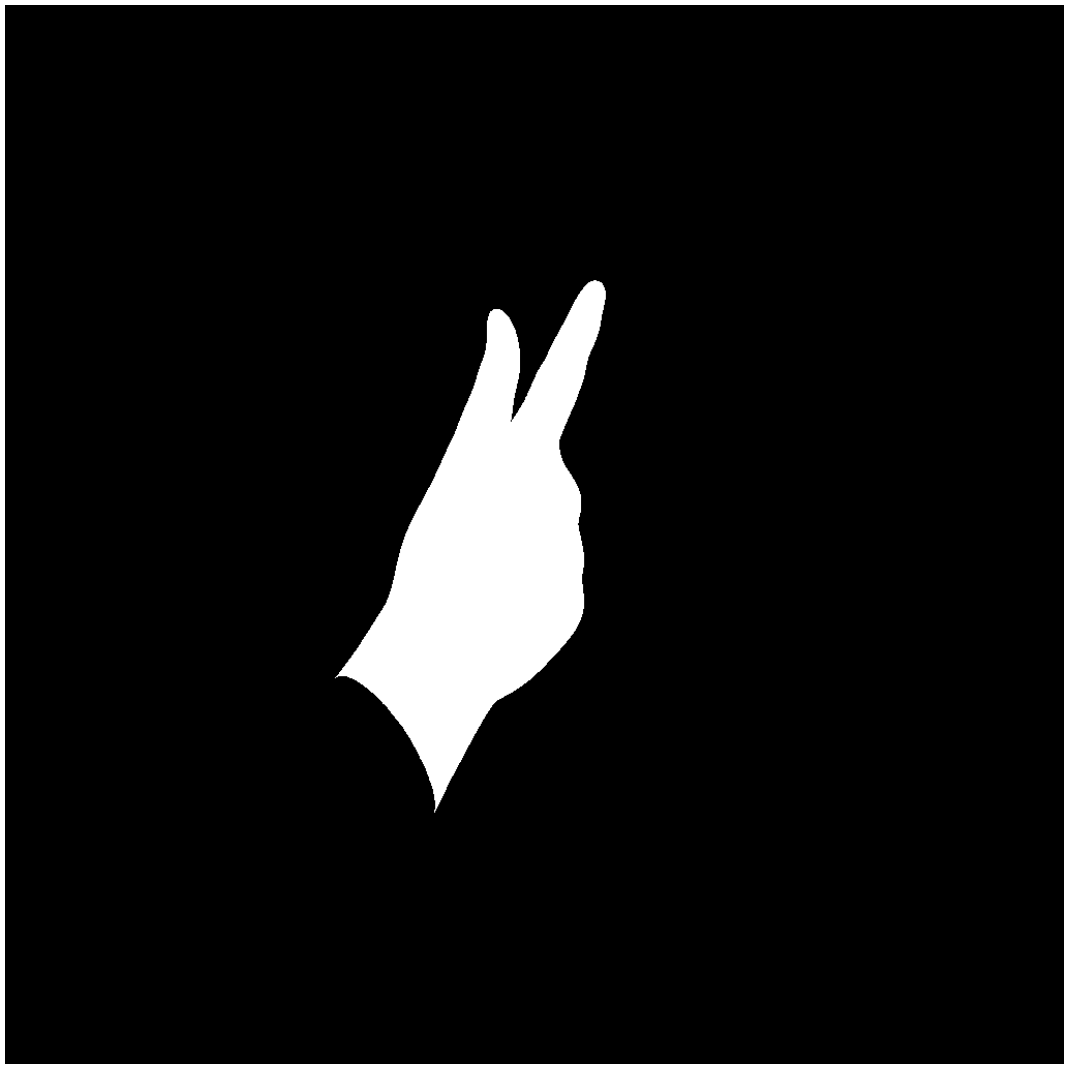


Figure 46. Showing mask #1 (culling and single-sided surface of model)

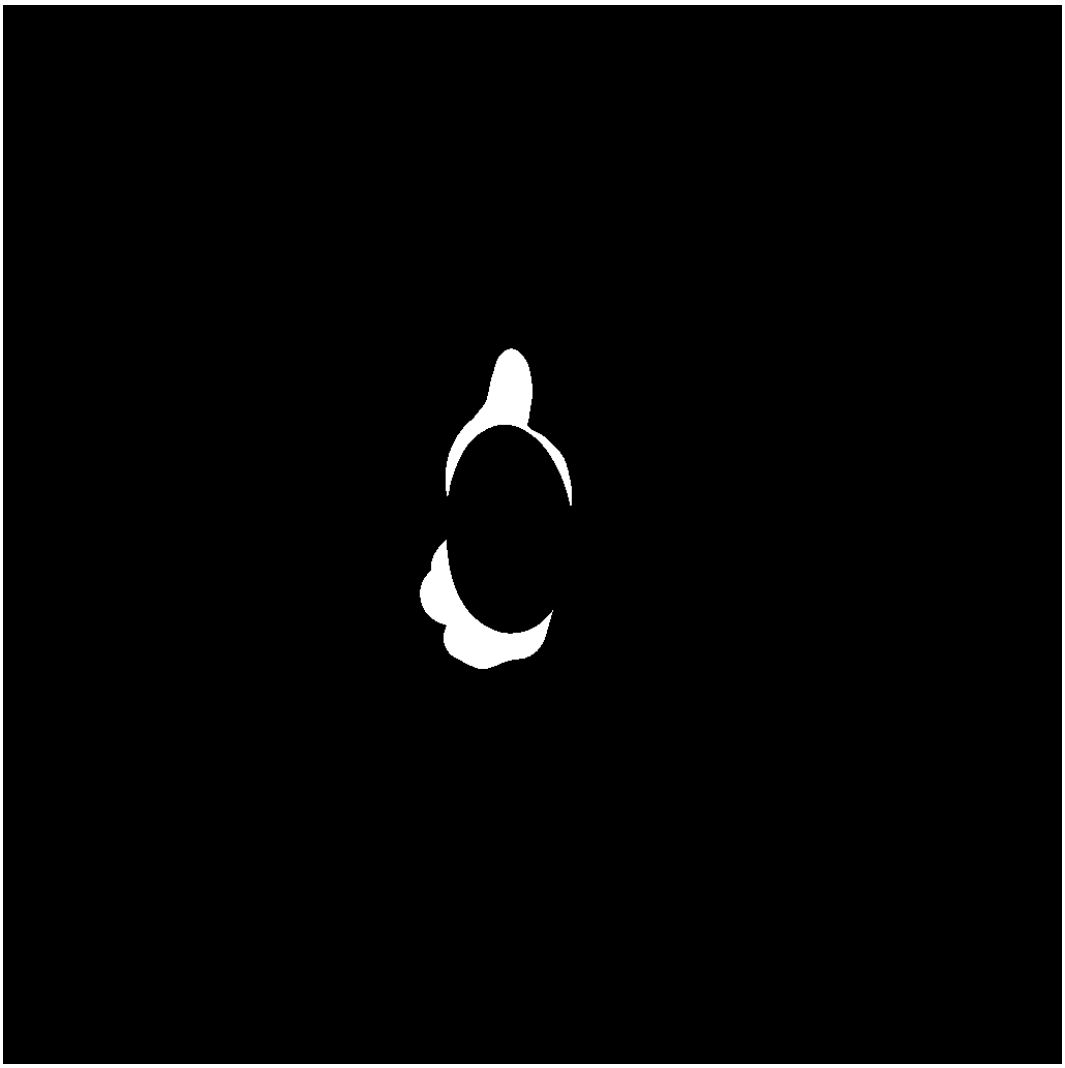


Figure 47. Showing mask #2 (culling and single-sided surface of model)

Similar to work done where color texture patches denoted hand pose, a simple texture map with colored fingers over a textured background was produced. This can be seen in Figure 48. However, it was determined that this might lead to too much extra texture information, and thus a flat colored texture was then used, as seen in Figure 49.

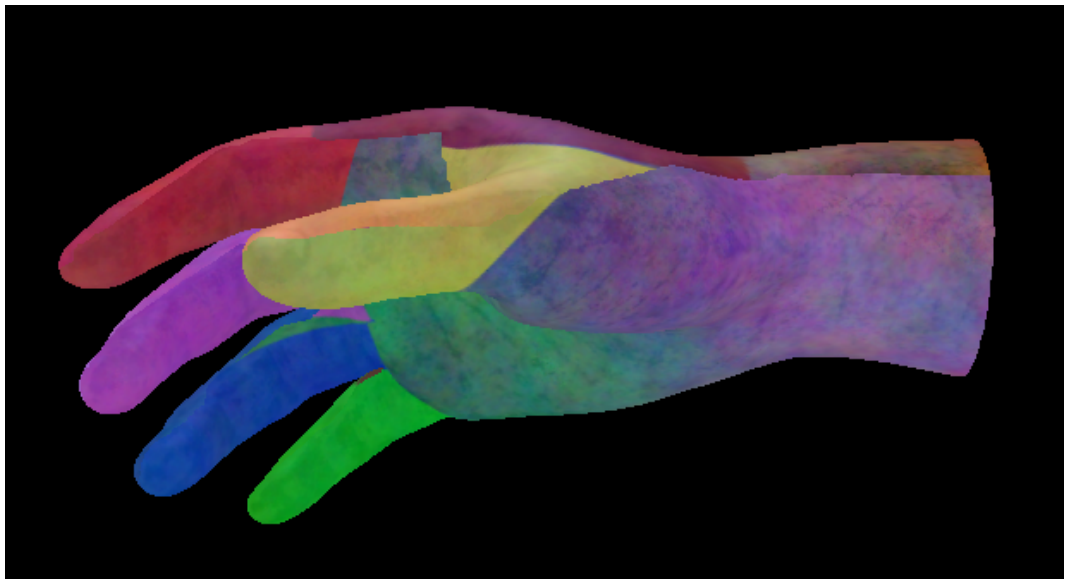


Figure 48. Simple hand-made per finger texture map.

Figure 49. Rendered per finger texture map as fed into the classifier system.

4.1.4 Phase 4: Rigid Objects

Presented here are the results of experiments described in Phase 4 (Subsection 3.2.4).

To understand the input for this stage of the project, Figure 50 is presented. It is an example of the set of images generated for training. Figure 50 is a 1000 by 1000 pixel image used as a source image, which is then re-sampled via 2D scaling to a standardized 224 by 224 pixels to fit Neural Network input layer as matching the standard.

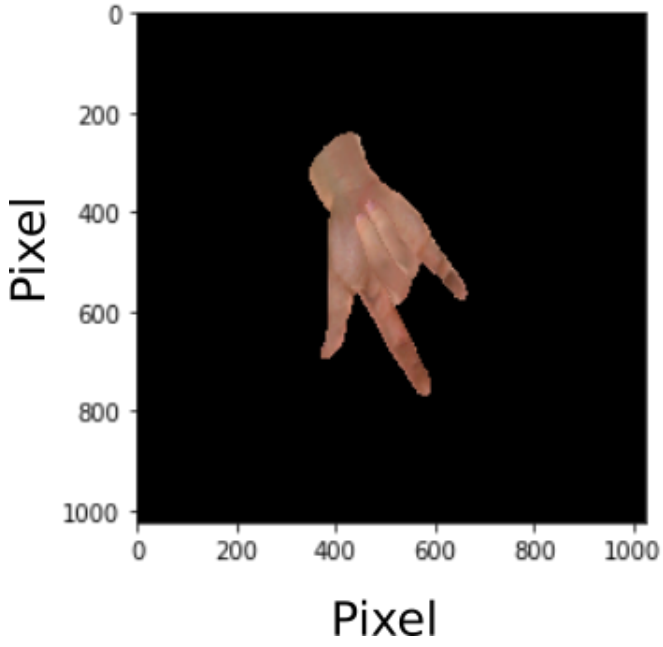


Figure 50. Example image input to the NN. Showing the "Rock On" pose.

4.1.5 Phase 5: Deformable Objects

Presented here are the results of experiments described in Phase 5 (Subsection 3.2.5).

To generate the first set of data for this phase, the rendering program was configured to generate images for the identified LibHand postures with rotation increments of 10 degrees for all angles across the three Cartesian axis. This was

done with the default texture map to give a baseline for evaluating the more complex hierarchical systems based on this rendering and neural network system. A simple neural network classifier was then trained on each pose as its own class. Then a test set was generated from the increments of 10 degrees and compared against the newly generated classifier. Further, several images of hand postures for each category were generated from real life using a low-end laptop webcam and also tested against the system. Ground-truth for this data set only includes the intended posture and does not include 3D point cloud data.

Confusion Matrices

The experiment included confusion matrices for training for 8 categories, one trained and validated with 500 images in each category, and the other trained and validated with an additional 500 images, yielding a total of 1000 images.

The first set of results are shown in Figures 51 and 52.

Note that images are 1024 pixels by 1024 pixels, and selected with uniform random distribution from the set of rotations of each postures where x, y, z rotations are multiples of 10 degrees.

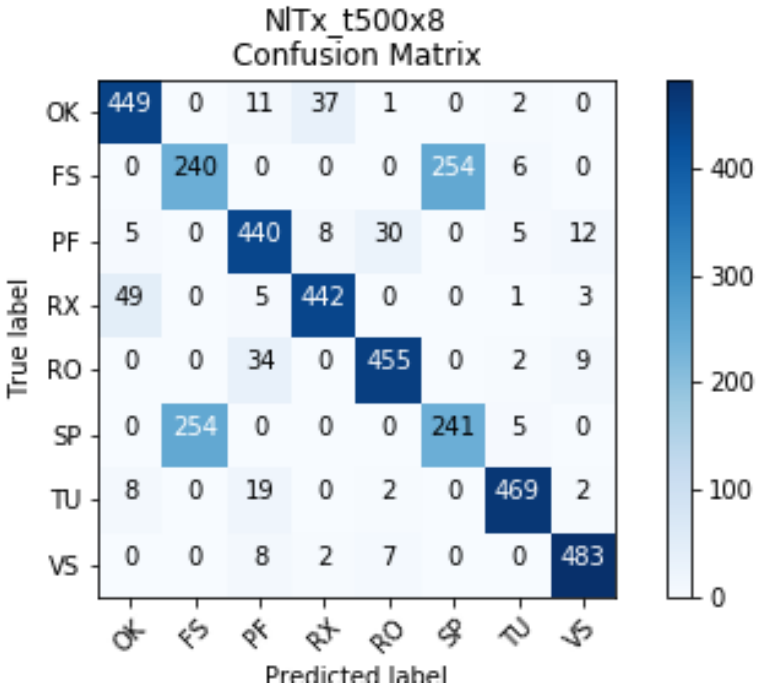


Figure 51. Experiment NlTx_t500x8: Confusion matrix of system with default LibHand texture with 500 images per class and all 8 default LibHand postures,

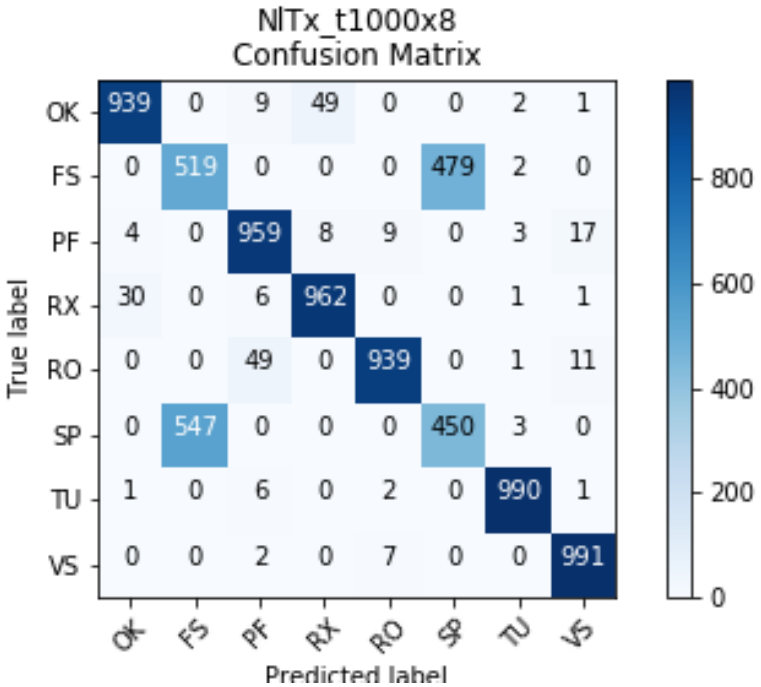


Figure 52. Experiment NlTx_t1000x8: Confusion matrix of system with default LibHand texture with 1000 images per class and all 8 default LibHand postures,

Varying the Number of Postures

To get a better understanding of the system's ability to be trained in general, a subset of the dataset was used to train a new model. Only the first four hand postures were used as possible categories. The system's results are shown in Figure 53.

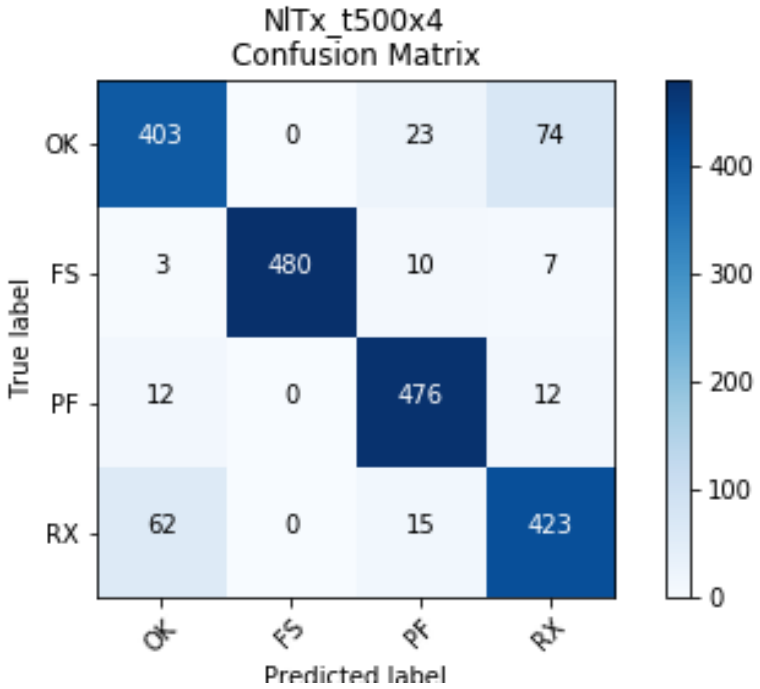


Figure 53. Experiment NlTx_t500x4: Confusion matrix of system running on the first four hand postures.

Next, more postures were added. The results from runs with only these new postures are shown in Figures 54 and 55, with 500 and 1000 training set versions, respectively.

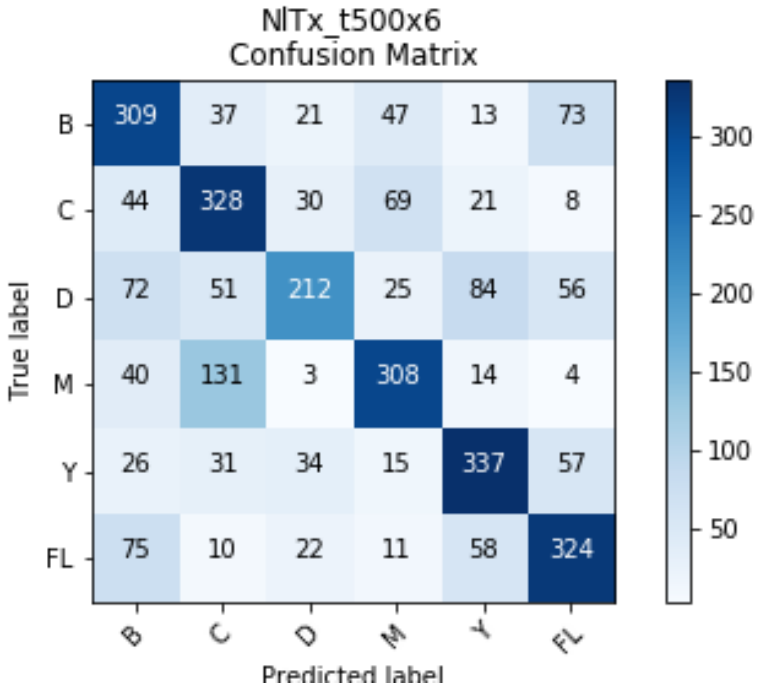


Figure 54. Experiment NlTx_t500x6: Confusion matrix of system running with 500 images per class, using hand signs for B,C,D,M, Y, and a hand held out flat.

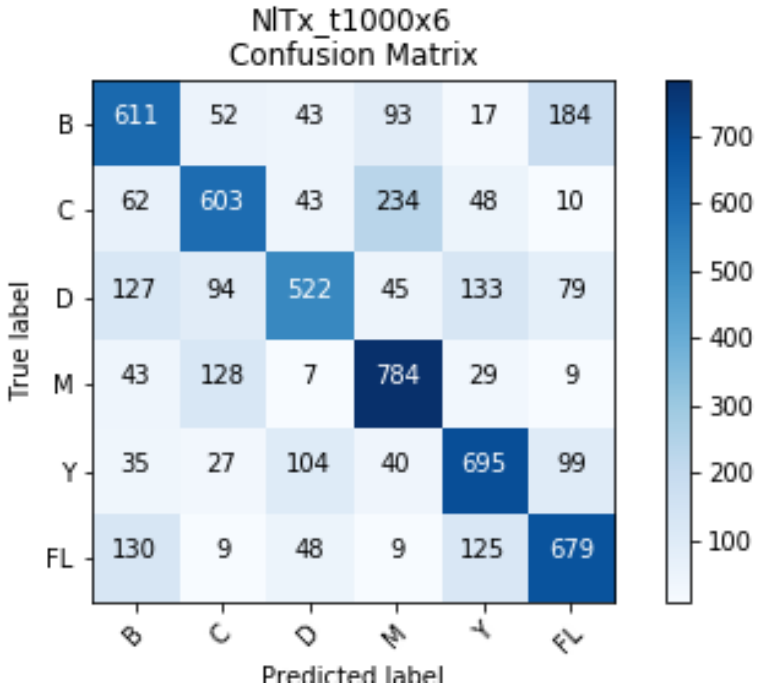


Figure 55. Experiment NlTx_t1000x6: Confusion matrix of system running with 1000 images per class, using hand signs for B,C,D,M, Y, and a hand held out flat.

Texture Masks and Noise

In addition to posture number comparisons, various textures were also compared with the static postures as classifier categories. The masks textures as generated earlier were used to train another NN. After seeing the results which were, surprisingly high given the lack of inside texture information, the system was tested against a non-masked validation set. This resulted in noticeably poor performance as was expected. The results from these two runs are shown in Figures 56 and 57.

Unlike most runs, which each took a considerable length of time on the available hardware, the masked runs finished in a reasonable amount of time, and were able to complete without running out of GPU memory, not only at small batches but at larger sizes as well. Previously, the hardware had run out of space after the data set started to run higher than 1000 per category with 14 categories. Although the system ran out of memory with 14 categories, it was successfully completed with 8. For this reason, the system was then trained again on the entire possible dataset of the default LibHand postures minus 500 per category for validation.

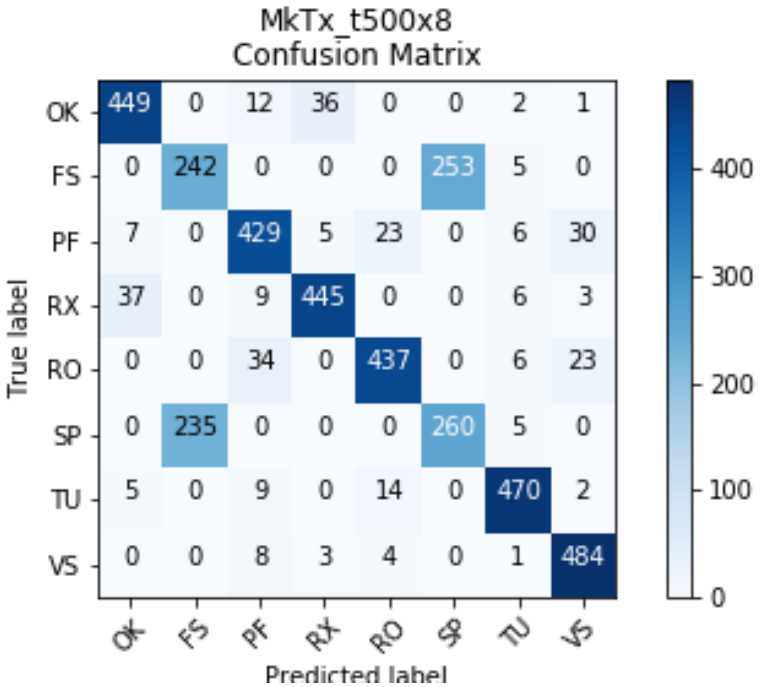


Figure 56. Experiment MkTx_t500x8: Confusion matrix of system running on a masked version of 8 postures, with 500 images per class.

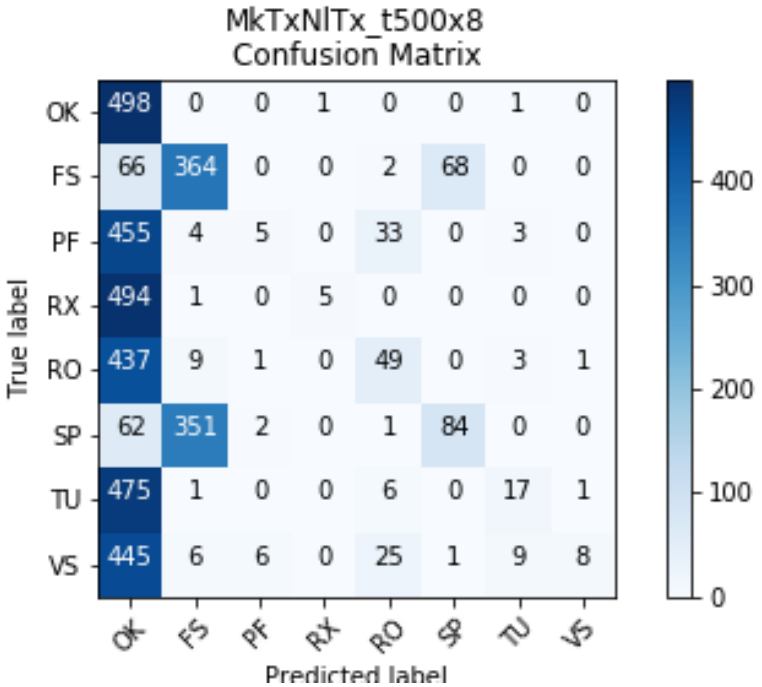


Figure 57. Experiment MkTxNlTx_t500x8: Confusion matrix of system running on default texture mapped posture renders of 8 postures, with 500 images per class, but having been trained on the masked data set.

Following the masked data experiments, random Red, Green, and Blue channel noise was explored, or just RGB noise. Similarly to the quest to understand how the contour affected the results, the next step was to develop intuition into the effect of the texture. Starting with a white texture map layer, RGB noise was added randomly. This was chosen because white would be the maximum difference between the segmented background in the render. To determine if using white and then applying RGB noise made it easier to identify against the black background, another set of training data based on a black layer with RGB noise was also conducted. Each was completed with 500 and 1000 training examples per category. The results from these four tests are shown in Figures 58, 59, 60, and 61.

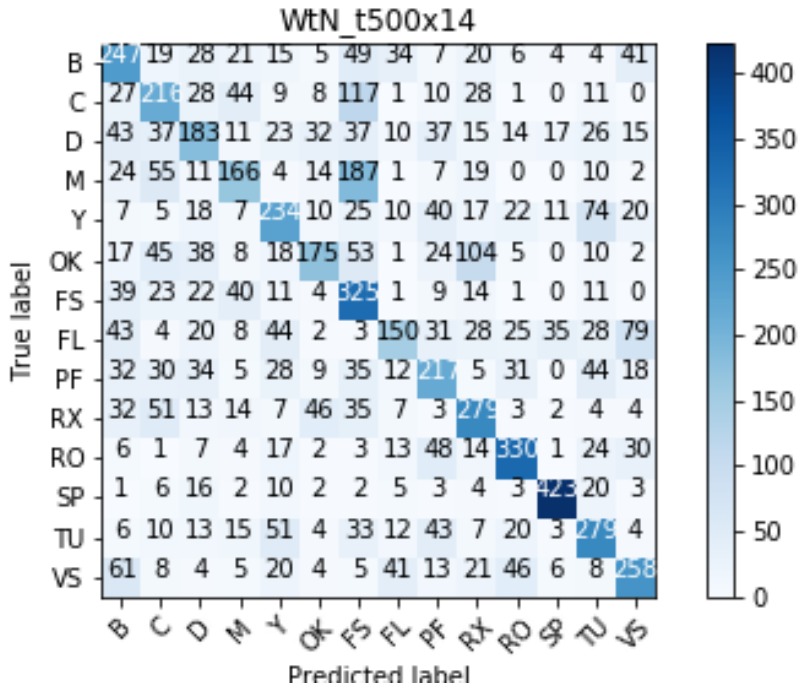


Figure 58. Experiment WtN_t500x14: Confusion matrix where system used the WtN texture map, 14 postures, and 500 images per class.

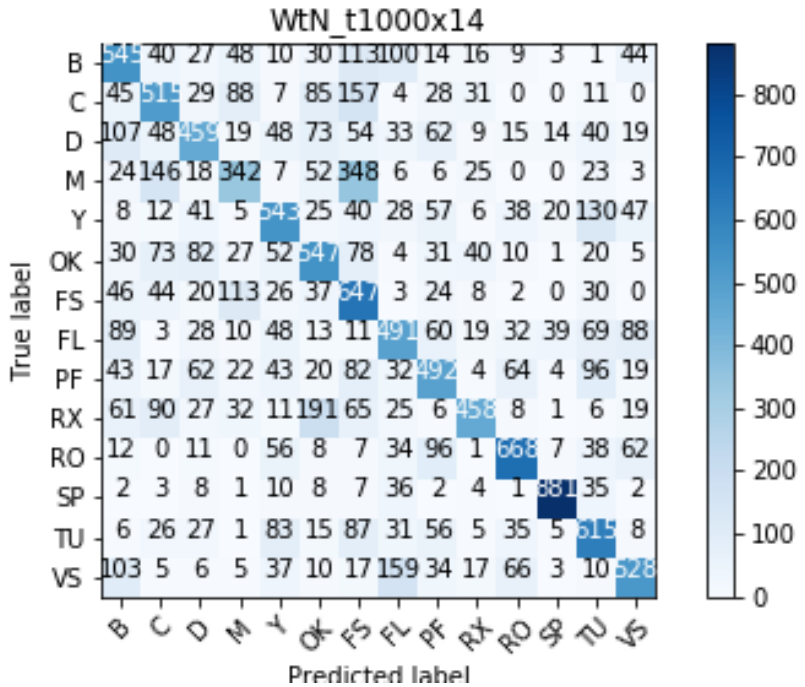


Figure 59. Experiment WtN_t1000x14: Confusion matrix where system used the WtN texture map, 14 postures, and 1000 images per class.

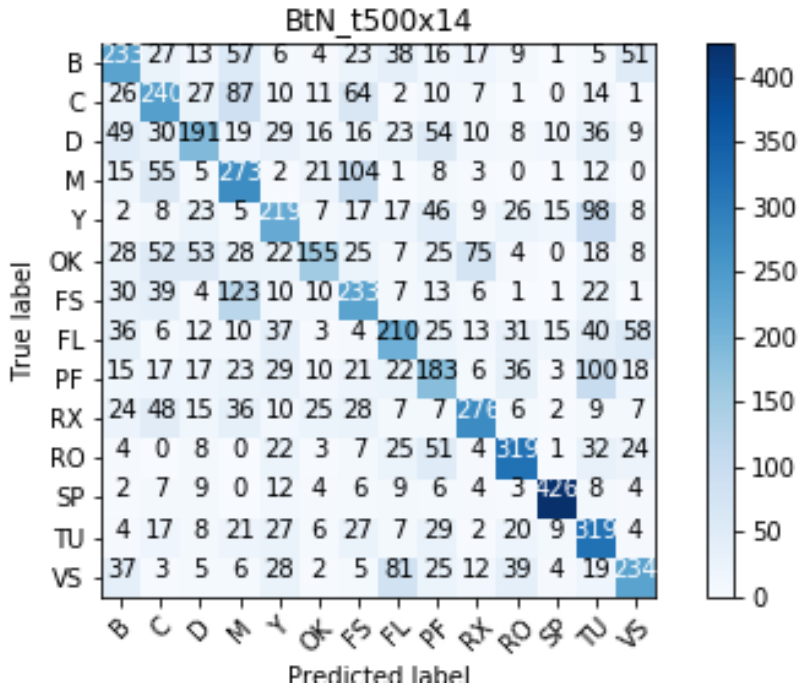


Figure 60. Experiment BtN_t500x14: Confusion matrix where system used the BtN texture map, 14 postures, and 500 images per class.

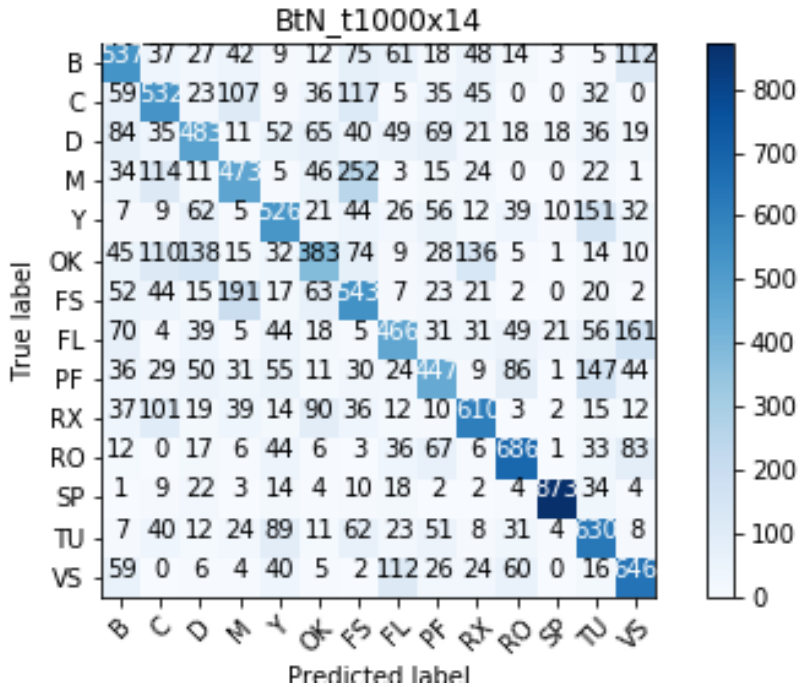


Figure 61. Experiment BtN_t1000x14: Confusion matrix where system used the BtN texture map, 14 postures, and 1000 images per class.

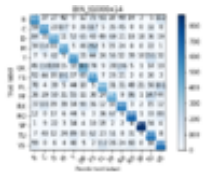
4.1.6 Phase 6: Noise

Presented here are the results of experiments described in Phase 6 (Subsection 3.2.6).

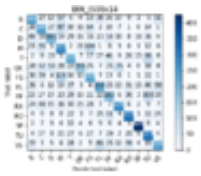
The effect of noise was deemed outside the project scope during this phase. It is left to future work.

4.2 Comparison of Models

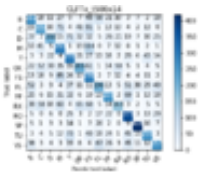
In the previous models tested, variables were the number of postures, training set size, and texture maps. At this point a meta-analysis of the models was conducted. Figure 64 shows selection error and accurate detection for the trained reader programs. Figure 65 shows the Indicated Accuracy for a couple of textures. Figure 66 shows the Indicated Accuracy for the default LibHand postures. To have an idea of how the number postures influenced the system, the models which used the default texture map were compared. Figure 67 presents these findings. The increase in postures appears to lower accuracy, but the inclusion of only confusing postures, such as the models trained on six categories, lowered the accuracy more than when trained on a full eight categories. This may be because of the larger training set overall, but it might also be partially caused by the inclusion of categories which were easier to classify, thus bringing up the overall accuracy values. Figures 68 and 69 depict the overall accurate detection and selection error for each texture, respectively. It is interesting to note that the CLFTx, WtN, and BtN had closely similar results. CLFTx, which has each finger already segmented, was initially expected to do significantly better than those with random noise, such as BtN and WtN. More research is clearly needed to verify that this is not random noise. A more comprehensive study is thus suggested, and would be expected to have interesting and useful results.



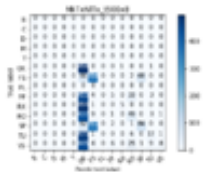
BtN_t1000x14
432x288



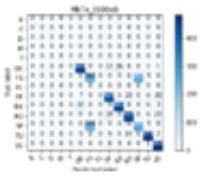
BtN_t500x14
432x288



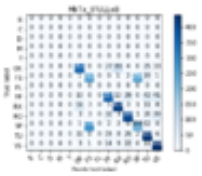
CLFTx_t500x14
432x288



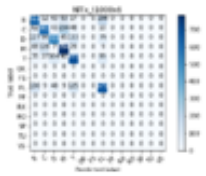
MkTxNlTx_t500x8
432x288



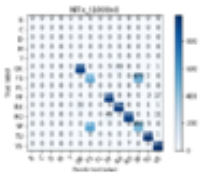
MkTx_t500x8
432x288



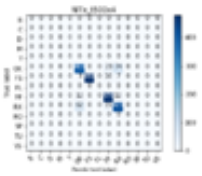
MkTx_tFULLx8
432x288



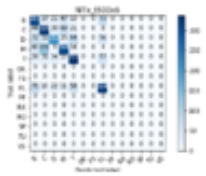
NlTx_t1000x6
432x288



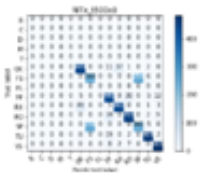
NlTx_t1000x8
432x288



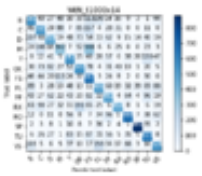
NlTx_t500x4
432x288



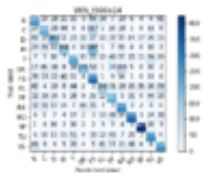
NlTx_t500x6
432x288



NlTx_t500x8
432x288



WtN_t1000x14
432x288



WtN_t500x14
432x288

Figure 62. Showing all confusion matrixes.

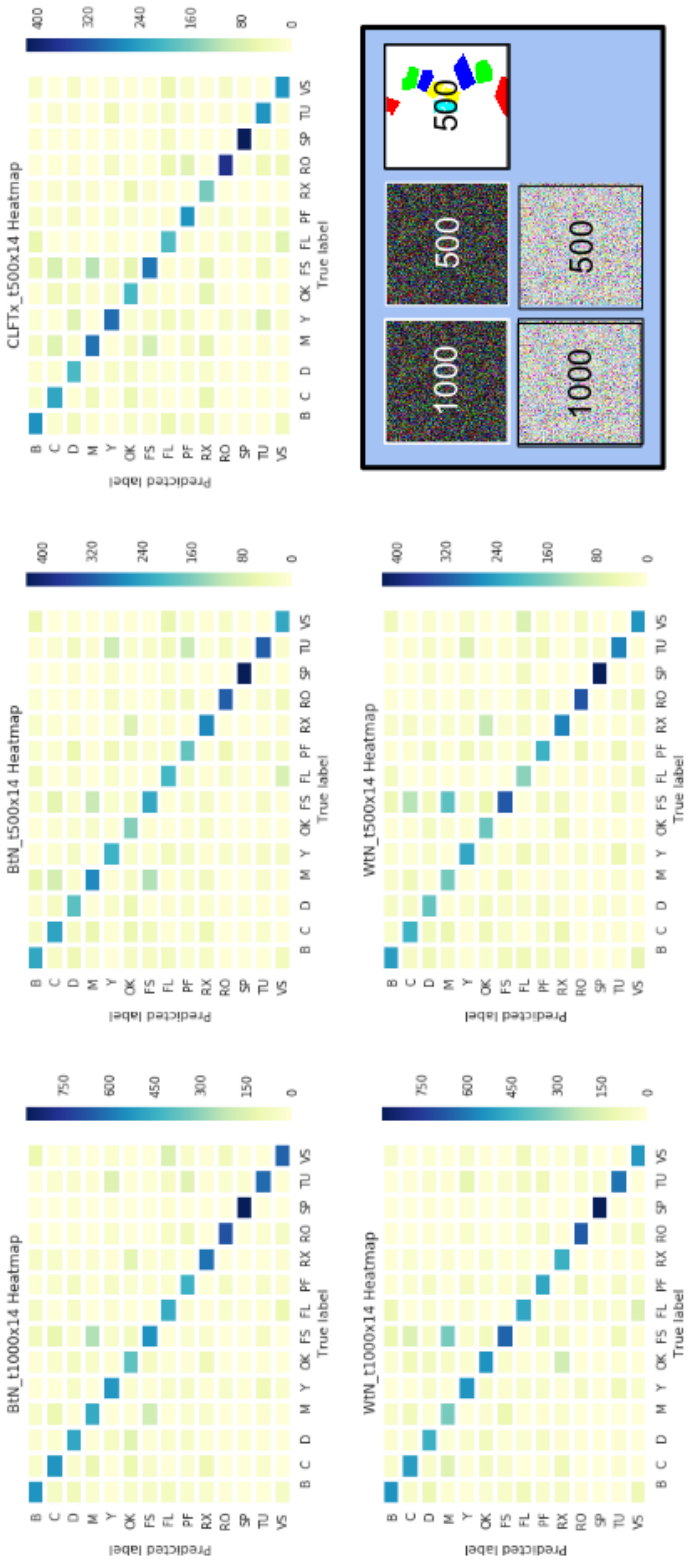


Figure 63. Heatmaps of runs with 14 postures

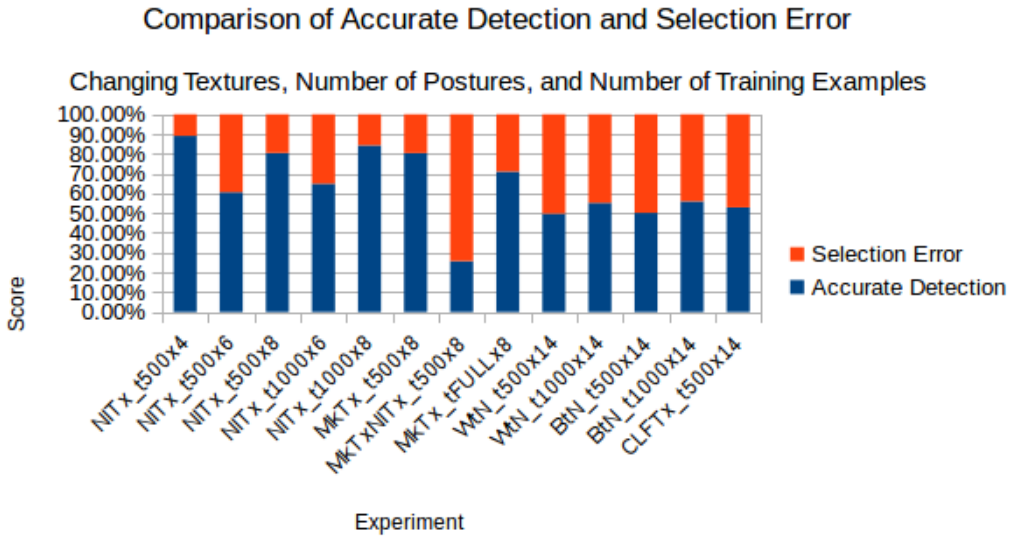


Figure 64. Comparison of accurate detection and selection error across posture category experiments.

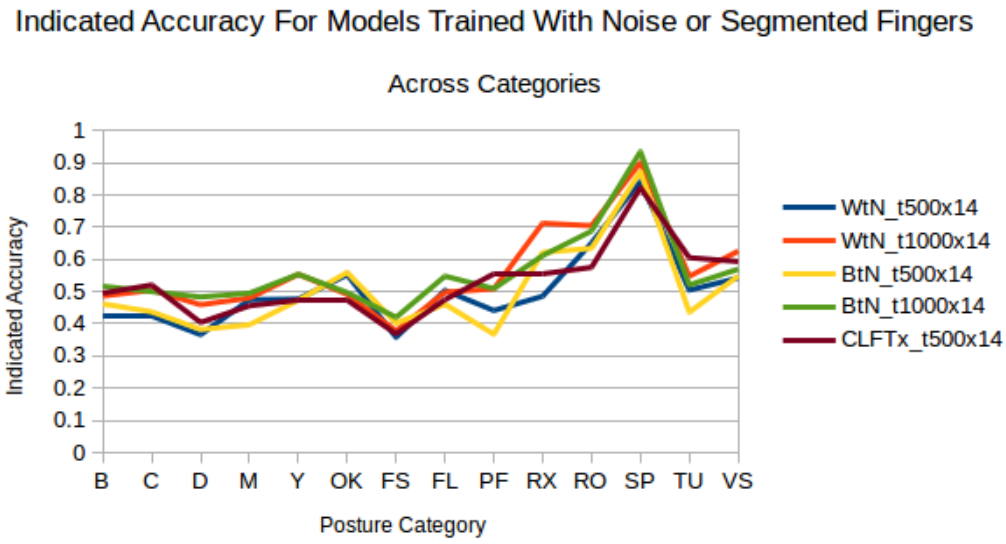


Figure 65. Indicated accuracy for noise and CLFTx textures.

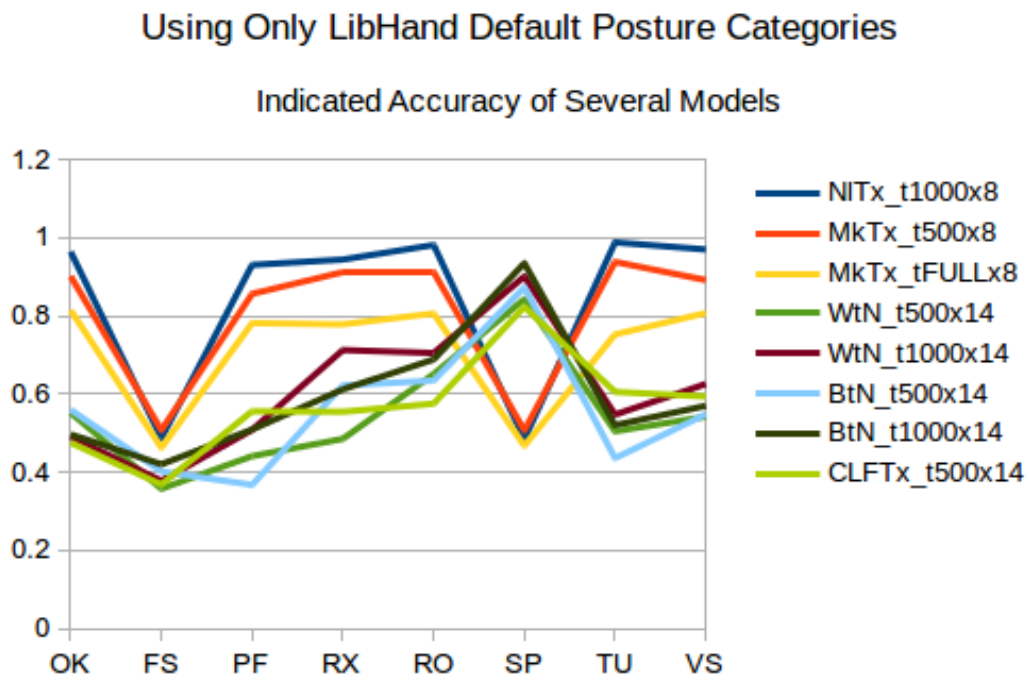


Figure 66. Indicated accuracy for default LibHand postures.

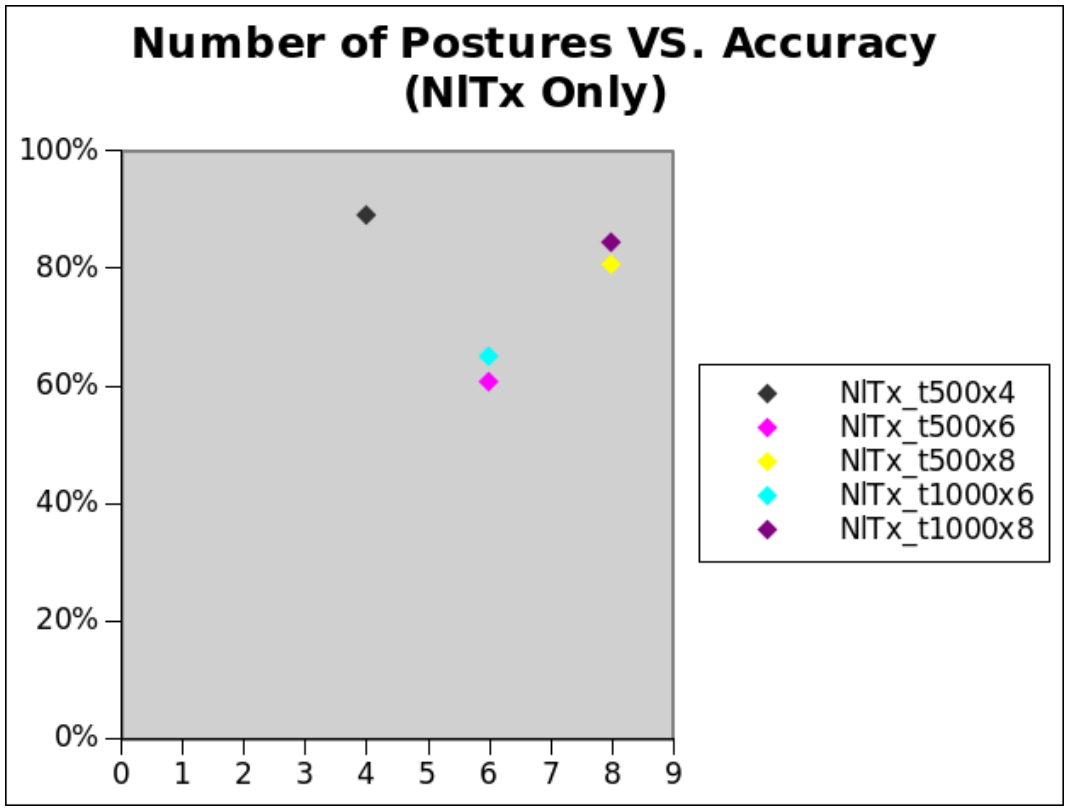


Figure 67. Comparison of number of postures versus accuracy.

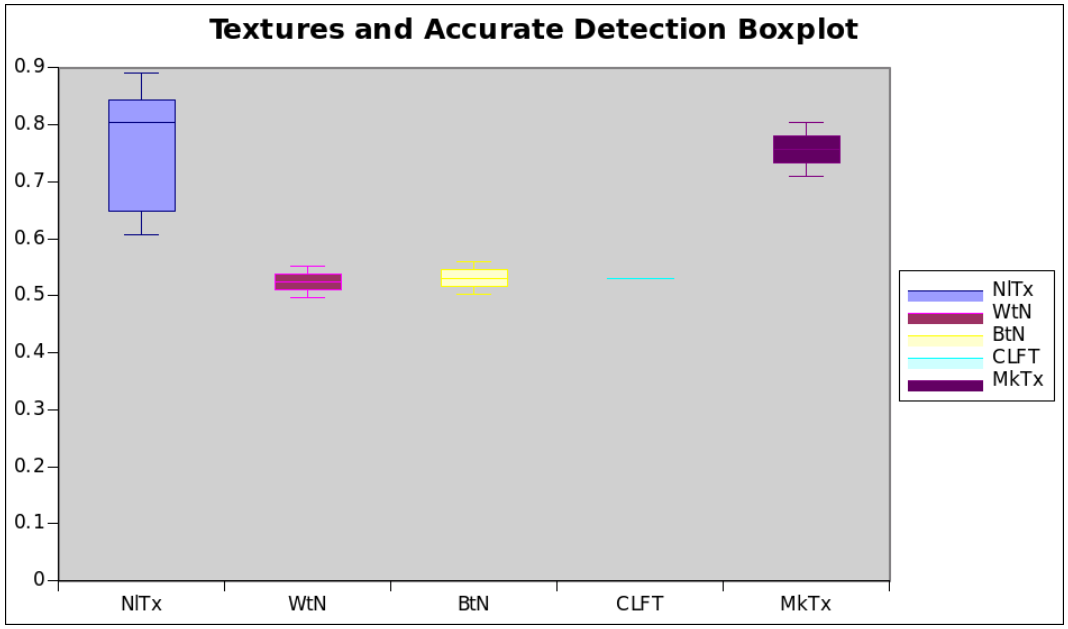


Figure 68. Textures and accurate detection.

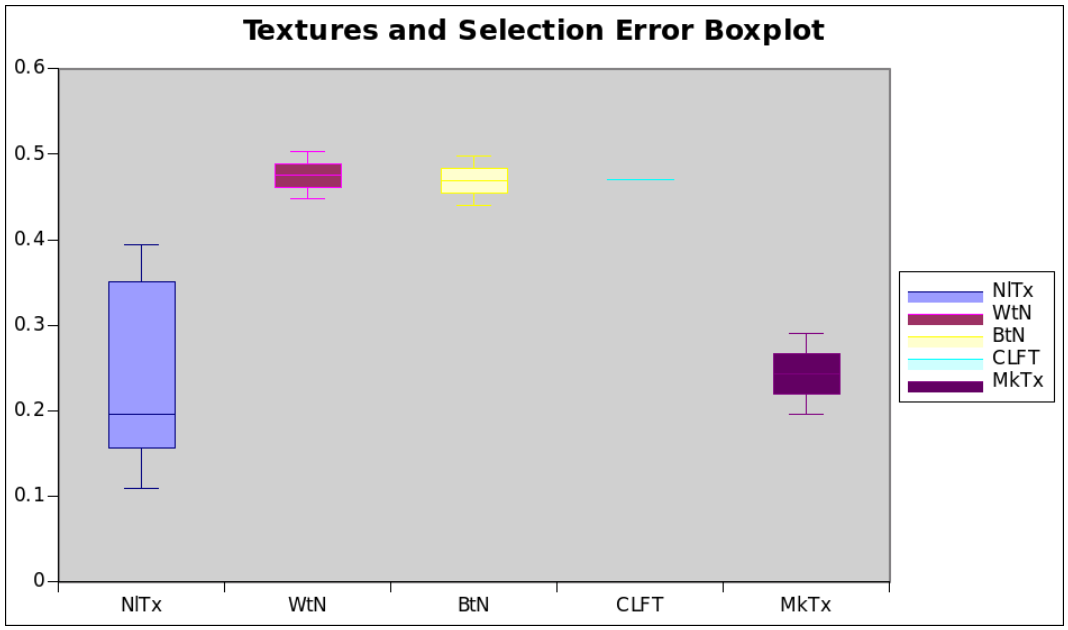


Figure 69. Textures and selection error.

List of References

[1] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? Visual explanations from deep networks via gradient-based localization," *CoRR*, vol. abs/1610.02391, 2016. Accessed on April 15, 2018. Available: http://arxiv.org/abs/1610.02391

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

This project begins the path for further development and exploration. As this was developed as part of a Master's program level project, the work resulted in both answering some questions, but opening up many more. New questions, such as the exact extent to which contour plays a role versus the role of inside texture, optimization problems in a system such as the one presented herein, improved designs for the tree structure of the genetic programming system, and many more are left to future readers.

This project is a viable solution to generate surface textures for digital reconstruction of the inverse-mapping problem of computer vision. This was done using a combination of three biology-inspired artificial intelligence techniques. Significant research opportunities remain in this field. Further work will be necessary to expand and leverage this project.

Several several different underlying libraries were selected based on their availability on POSIX compliant computers and their licenses. It was interesting to find that during the execution of this project, large swathes of projects which were considered as potential candidates but at the time ruled out due to these types of issues, were re-released either with better licenses and/or for GNU/Linux.

The supported version of Ogre3D available on the platform to which I had access also moved licenses during this project's duration.

Although ZeroMQ was chosen in the end, when NanoMSG becomes more developed, it might be beneficial to revisit.

One of the things that did not work well was the duration of this project. This project's scope was found to be much larger than expected, and as experiments

were conducted and results were analyzed, the questions and complexity of the system increased. The extended timeline itself caused cost overrun as well as additional concerns. Of particular note was that the project's extended duration caused difficulty maintaining completed modules. Despite these modules being fully implemented, on returning to them after an extended intermission, many of the supporting libraries had undergone change or become unmaintained such as to necessitate further work before using them again.

5.2 Future Work

Although substantial software was written, the field remains open and more work is needed to further explore what was learned in this thesis.

Although it was not done during the creation of this thesis, a logical next step for a continuation of this work would be to compare the results against people. Here, either photographs or video of a real person's hand could be fed into the system.

Because there are newer systems which do not necessarily require a glove, it would also be interesting to compare them against this one. We could print the glove texture on a physical glove and have participants enact various pose, posture, and gesture combinations. Furthermore, we could have a 3D point cloud being generated at the same time as a way of obtaining the ground-truth. Both gloved and non-gloved results would then be compared.

The exact extent to which this system is suitable for deployment as an input system for use with low-power, low-resource, real-time embedded devices is not yet thoroughly explored. However, the current state of the research indicates high likelihood that the techniques that were explored would lend themselves to be a good fit for this purpose. Likewise, the extent to which this is applicable for other three-dimensional surfaces requires more work, but the results so far are

very encouraging.

As other possible future research, if they were to be addressed, it might be done by incorporating simulated inter-frame time-based errors into the training dataset. Various other issues often observed in sequential images could potentially be investigated and possibly simulated, such as motion blur, frame loss, and image artifacts. By training the AI on these simulated noise datasets, it is believed that issues that may otherwise arise from unforeseen complications related to the intrinsic problems of attempting to sample continuous imagery with discreet polling may be avoided.

APPENDIX A

Technical Resources

A.1 Writing Tools of the Thesis

This document was written using the document generation programming language known as LaTeX[1], while the project proposal was written in Lyx, an editor and associated higher-level abstraction layer which sits on LaTeX. The structure of this document is based on the requirements set by the University of Rhode Island for theses and uses one of their approved templates [2, 3]. Resources such as books describing and advising on the thesis and dissertation process were also used [4]. A large portion of the thesis was written directly in LaTeX, directly into the online Overleaf editor. Charts not generated directly out of the code were made in Open Office. Figures which show multiple rendered images were joined together using ImageMagick. Custom diagrams, such as those showing program flow and made as vector art, were produced using a combination of Gimp and Inkscape.

A.2 Software

A.2.1 Computer Systems

All software was developed on GNU/Linux compatible systems. This was chosen due to their availability, standardization, reliability and documented implementation.

A.2.2 Programming Languages

Several programming languages were used for this project. The software initially used for exploring the 3D model and texture map combination was written in Java using the Processing library extensions. At this time Java and the R programming languages were being heavily pushed by the Computer Science department in the graduate curriculum. Therefore, the complimentary graphing software written

at this preliminary phase was R based.

These are presented below:

- Java
- C
- C++
- Pascal
- Python
- CUDA
- Bash
- R

Note that CUDA is considered a language here and not just a library, because it has its own specialized hardware, compiler, debugging, and profiling toolchain, even if it is a modified version of C++. Moreover, this distinction from a library also applies to C++ from C, where the former was originally simply a library wrapper for the latter.

A.2.3 Programming Libraries

While the system was originally designed from the ground up using homemade components, during the lifetime of the project several tools across the entire toolchain were made available and were thus incorporated into the project. Eventually Python was used to properly tie everything together, leaving only the renderer as a non-Python tool, still written in C++.

Supporting software such as the open source or free software libraries LibHand [5, 6], OpenGL [7], Ogre3D [8], OpenCV [9], CUDA [10], and OpenMP [11], were used. In the original GA software OpenMP in CUDA were used for multi-computer / multi-CPU and GPU acceleration. Boost is used to handle tasks such a sorting, file system management, and several other key tasks which are not normally features of C++. Basic image processing in this version was handled by a combination of OpenCV and Allegro. 3D graphics in C++ were written using Ogre3D. This was chosen because the 3D hand model system, LibHand uses it internally and it made integration easier.

The GA system was later rewritten using the Deap library for Python. Instead of using OpenMP, the later system used LibZMQ, which is an implementation of the ZeroMQ protocol standard, to handle the interconnection between the different modules.

Plotting and graphs were either created using Matplotlib or Scikit-Learn when produced in Python, or RGL when produced in R.

At the beginning of the project the selection of neural network libraries were limited. This originally resulted in a simplistic use of this powerful technology to mitigate the unavailability of this resource. However, as this project continued, library after library were made available to the average developer. Even high profile libraries such as TensorFlow and PyTorch, developed by companies such as Google and Facebook, respectively, followed this trend. These two libraries are high-level machine learning tools which became available almost overnight. At that point the much more limited self-made system I had been writing at the time was instantly dropped in favor of an indisputably better tool. TensorFlow at the time was chosen based on documentation support. The even higher-level Keras extension, which controlled PyTorch or TensorFlow, was eventually incorporated into TensorFlow, and once officially supported, was explored as a possible resource for this project. As a result, initial NN work was done using TensorFlow/Keras.

Towards the end of the project the NN system was found to be running too slowly on the available hardware.

Based on a fellowship received which sponsored attendance in a remotely taught course on the subject, the FastAI library, an alternative to Keras which was used in the course, was tested and found to resolve some of the speed deficiencies. The underlying system of FastAI is PyTorch, and the underlying system of all of these neural net tool chains is cuDNN written in C++/CUDA. This remains

the underlying structure for almost all of the higher level libraries available for this task due to speed optimizations, even though the higher-level tools are often preferentially accessed via Python.

A.3 Hardware

The pre-thesis literature review phase was almost started on an outdated MacOS device, but due to user permission issues and the inability to log into it, this system was rendered useless. Without permissions to install a compiler or similar computer science tools, it was abandoned. A more readily usable early 2000s Athlon CPU computer was used instead, which had GNU/Linux installed on it.

It was not expected that resources beyond those provided by the University were required. However, since this project was computationally expensive, the hardware used needed to be fast and specially designed for parallel execution. To complete this research project considerable time, access to a high-powered computer (HPC), and possibly a variety of different types of digital cameras, such as the webcams included on most modern laptops will be helpful. While computation can be performed on a standard personal computer, access to an HPC will provide more meaningful results within the given time-frame. Access to the shared, modern HPC for Rhode Island universities was not granted during this project, although minimal access to the old server, which only had an out-dated and relatively unuseful Fermi-based NVIDIA GPU architecture, was granted. Unfortunately, this was not sufficient to run the current generation of algorithms, and thus a home machine was procured, built, administrated, and used.

Several different primary machines were used. The main machine used for the intensive computations was a GPU-based personal computer equipped with an early CUDA-capable NVIDIA card. Later on an NVIDIA GTX 960 was used, and

towards the end of the project it was distributed on either an GTX 1080 or GTX 1050.

List of References

[1] L. Lamport, *LATEX: a document preparation system.* Addison-Wesley Pub. Co., 1986. Accessed on April 15, 2018. Available: https://books.google.com/books?id=IgJUAAAAMAAJ

[2] T. M. Toolan *et al.* University of Rhode Island. "Guide to Writing Your Thesis in LaTeX." Accessed on April 15, 2018. June 2006. Accessed on April 15, 2018. Available: https://web.uri.edu/ecbe/thesisguide/

[3] T. M. Toolan and D. W. Tufts, "Detection and estimation in non-stationary environments," in *Proceedings IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.

[4] K. L. Turabian, *A Manual for Writers of Term Papers, Theses, and Dissertations, 6th. edn.* Chicago, Illinois, United States of America: University of Chicago Press, 1987.

[5] E. Shasheen, M. Šarić, *et al.*, "Libhand: A library for hand articulation," 2014, version 0.9.z. Accessed on April 15, 2018. Available: http://www.libhand.org/

[6] M. Šarić, "Libhand: A library for hand articulation," 2011. Accessed on April 15, 2018. Available: http://www.libhand.org/

[7] Silicon Graphics, Khronos Group, *et al.*, "OpenGL (Open Graphics Library)." Accessed on April 15, 2018. Available: https://www.opengl.org/

[8] The OGRE Team *et al.*, "OGRE (Object-oriented Graphics Rendering Engine)." Accessed on April 15, 2018. Available: https://www.ogre3d.org/

[9] Intel Corporation, W. Garage, *et al.*, "OpenCV (Open source Computer Vision)." Accessed on April 15, 2018. Available: https://www.opencv.org/

[10] NVIDIA Corporation, "CUDA (Compute Unified Device Architecture)." Accessed on April 15, 2018. Available: https://www.nvidia.com/object/cuda_home_new.html

[11] OpenMP Architecture Review Board *et al.*, "OpenMP (Open Multi-Processing)." Accessed on April 15, 2018. Available: http://openmp.org/

APPENDIX B

Additional Figures

B.1 Rigid_EC

Preliminary data from the Rigid_EC system is presented in Figure B.70. It is too early, at the time of this writing, to make certain statements concerning the trajectory of the system's indicated texture map accuracy. However, it does suggest further research may be worthwhile. Figures B.71 and B.72 show two confusion matrices from the best individuals during the limited run of the automated system. The categories of the this NN system are the octants a 3D model could be facing. Figure B.73 shows the system running.

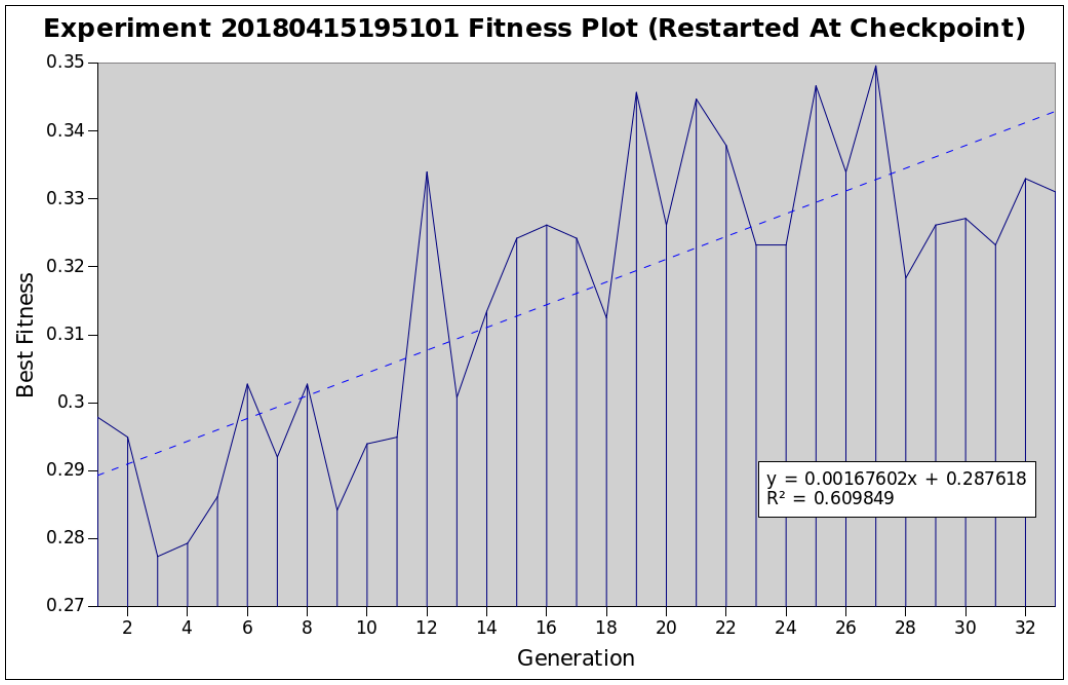


Figure B.70. Current ongoing run of the Rigid_EC at the time of this writing.

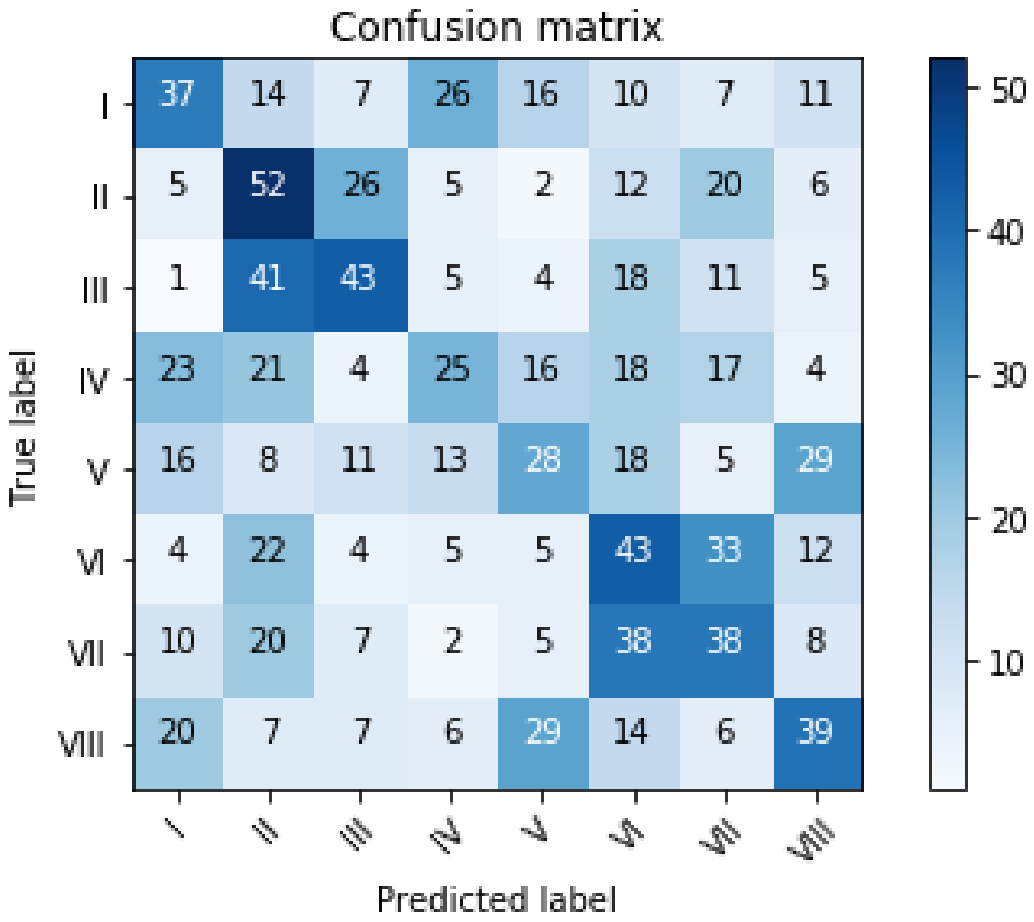


Figure B.71. Rigid_EC run: Confusion matrix of best individual at beginning.

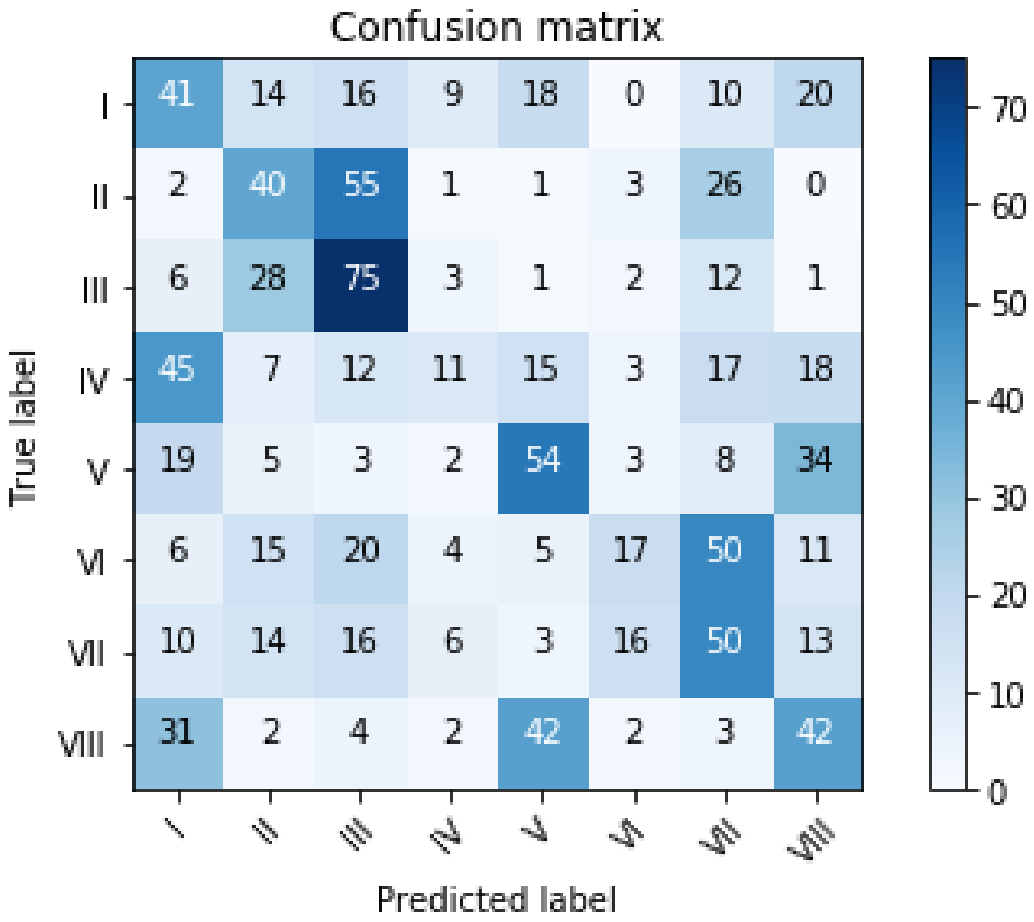


Figure B.72. Rigid_EC run: Confusion matrix of best individual at a later generation.

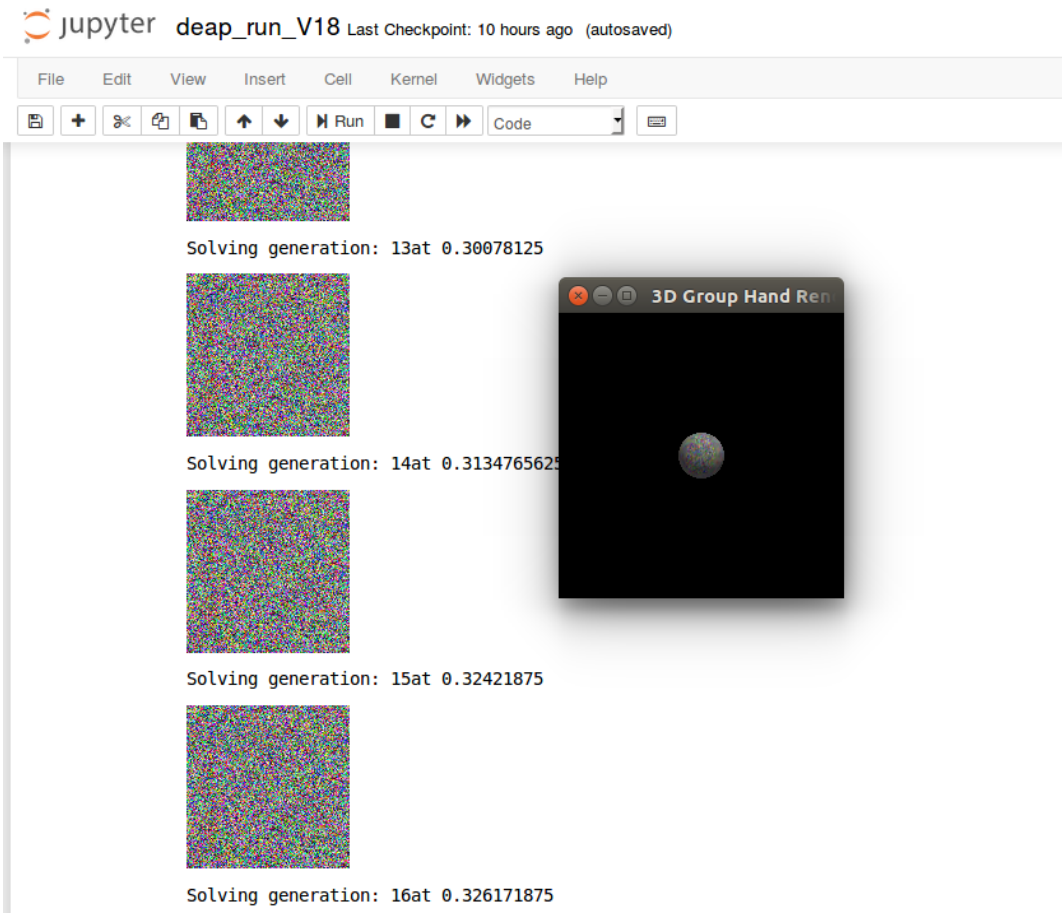


Figure B.73. Running the system on a sphere mesh

APPENDIX C

Further Notes

C.1 Note: Structure

This Master's thesis is written in a format as prescribed by the University of Rhode Island (URI). The organization of the chapters is based on the standard for URI.

The structure is that of:

- Chapter 1 — High-level introduction to the topics discussed herein.
- Chapter 2 — Information necessary to understand the background of the research, for those not yet familiar with the work.
- Chapter 3 — Experimental design and reasoning.
- Chapter 4 — Experimental results and their interpretation.
- Chapter 5 — Conclusions and opportunities for future work.

C.2 Note: LibHand License and Citation

Many of the software used to produce this are released as open source or under similar licenses. Please check the appropriate project webpages for more detailed information. To meet the license requirements of LibHand, consider the the following its official citation in this work, and unofficial the in-text citation using a corrected bibtex entry and included with the LaTeX command `\cite`, which lists access date to meet URI's requirements. This is done as the license explicitly requires it to be presented in this format or another provided on its website. However at the time of this writing, no alternative format is provided.

The LibHand citation:

```
@misc{libhand,
  author = "Marin \v{S}ari\'{c}",
  title = "LibHand: A Library for Hand Articulation",
  year = "2011",
  url = "http://www.libhand.org/",
  note = "Version 0.9"
}
```

C.3 Note: Document Version

This version of the document is 2018-04-20.

BIBLIOGRAPHY

Abtahi, M., Constant, N. P., Gyllinsky, J. V., Paesang, B., D'Andrea, S. E., Akbar, U., and Mankodiya, K., "Wearup: Wearable e-textiles for telemedicine intervention of movement disorders," in *Wearable Technology in Medicine and Health Care*. Elsevier, 2018, [Expected publish date: June 1st, 2018]. Available: https://www.elsevier.com/books/wearable-technology-in-medicine-and-health-care/tong/978-0-12-811810-8

Abtahi, M., Gyllinsky, J. V., Paesang, B., Barlow, S., Constant, M., Gomes, N., Tully, O., D'Andrea, S. E., and Mankodiya, K., "Magicsox: An e-textile iot system to quantify gait abnormalities," *Smart Health*, 2017. Available: http://www.sciencedirect.com/science/article/pii/S2352648317300296

Albrecht, I., Haber, J., and Seidel, H.-P., "Construction and animation of anatomically based human hand models," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 98–109. Available: http://dl.acm.org/citation.cfm?id=846276.846290

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K., "Synthesizing robust adversarial examples," *CoRR*, vol. abs/1707.07397, 2017. Available: http://arxiv.org/abs/1707.07397

Ballard, D. H. and Brown, C. M., *Computer Vision*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc, 1982.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D., *Genetic Programming An Introduction On the Automatic Evolution of Computer Programs and Its Applications*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann, 1998.

Bovik, A. *et al.*, *Handbook of Image & Video Processing*, 2nd ed., Bovik, A., Ed. Burlington, MA, USA: Elsevier Academic Press, 2005.

Choi, E., Kim, H., and Chung, M. K., "A taxonomy and notation method for three-dimensional hand gestures," *International Journal of Industrial Ergonomics*, vol. 44, no. 1, pp. 171–188, 2014. Available: http://www.sciencedirect.com/science/article/pii/S0169814113001285

Dosselmann, R. and Yang, X. D., "A comprehensive assessment of the structural similarity index," *Signal, Image and Video Processing*, vol. 5, no. 1, pp. 81–91, 2009. Available: http://dx.doi.org/10.1007/s11760-009-0144-1

Du, K.-L. and Swamy, M. N. S., *Neural Networks and Statistical Learning*, 1st ed. NY, USA: Springer, 2013.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X., "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, 2007, special Issue on Vision for Human-Computer Interaction. Available: http://www.sciencedirect.com/science/article/pii/S1077314206002281

Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, 1st ed., ser. MIT Press. Cambridge, MA, USA: A Bradford Book, 1995.

Hillis, W. D., "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, pp. 228–234, 1990.

Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 1992.

Intel Corporation, Garage, W., *et al.*, "OpenCV (Open source Computer Vision)." Available: https://www.opencv.org/

Koza, J. R., Bennett, III, F. H., Andre, D., and Keane, M. A., *Genetic Programming III: Darwinian Invention and Problem Solving*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., 1999.

Lamport, L., *LATEX: a document preparation system*. Addison-Wesley Pub. Co., 1986. Available: https://books.google.com/books?id=IgJUAAAAMAAJ

Lathuilière, F. and Hervé, J. Y., "Visual tracking of hand posture with occlusion handling," in *Proceedings of the International Conference on Pattern Recognition*, vol. 3, Barcelona, Spain, 2000, pp. 1129–1133 vol.3.

Lin, J., Wu, Y., and Huang, T. S., "Modeling the constraints of human hand motion," in *Proceedings of the IEEE Workshop on Human Motion*, Austin, TX, USA, 2000, pp. 121–126.

McConnell, R., "Method of and apparatus for pattern recognition," Jan 1986, uS Patent 4,567,610. Available: http://www.google.co.uk/patents/US4567610

Mitchell, M., *An Introduction to Genetic Algorithms*, 1st ed. Cambridge, MA, USA: The MIT Press, 1996.

Newell, M. E., "The utilization of procedure models in digital image synthesis," Ph.D. dissertation, University Of Utah, UT, USA, January 1975, original source of the Utah Teapot also know as the Newell Teapot.

NVIDIA Corporation, "CUDA (Compute Unified Device Architecture)." Available: https://www.nvidia.com/object/cuda_home_new.html

OpenMP Architecture Review Board *et al.*, "OpenMP (Open Multi-Processing)." Available: http://openmp.org/

Ott, B., Gyllinsky, J. V., Jouett, N., Alashwal, H., and Martin, L. M., "Parameter optimization using genetic algorithms for namd - saving time on computational chemistry with an initial ga pre-processing step," 2017. Available: https://web.uri.edu/gradconference/

Pamplona, V. F., Fernandes, L. A. F., ao Prauchner, J., Nedel, L. P., and Oliveira, M. M., "The image-based data glove," in *Proceedings of the X Symposium on Virtual Reality*, SBC. Porto Alegre, RS: SBC, 2008, pp. 204–211.

Pavlovic, V. I., Sharma, R., and Huang, T. S., "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, Jul 1997.

Rempel, D., Camilleri, M. J., and Lee, D. L., "The design of hand gestures for human-computer interaction: Lessons from sign language interpreters," *International Journal of Human-Computer Studies*, vol. 72, no. 10–11, pp. 728–735, 2014. Available: http://www.sciencedirect.com/science/article/pii/S1071581914000706

Rhee, T., Neumann, U., and Lewis, J. P., "Human hand modeling from surface anatomy," in *Proceedings of the Symposium on Interactive 3D Graphics and Games*, ser. I3D '06. New York, NY, USA: ACM, 2006, pp. 27–34. Available: http://doi.acm.org/10.1145/1111411.1111417

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D., "Grad-cam: Why did you say that? Visual explanations from deep networks via gradient-based localization," *CoRR*, vol. abs/1610.02391, 2016. Available: http://arxiv.org/abs/1610.02391

Shasheen, E., Šarić, M., *et al.*, "Libhand: A library for hand articulation," 2014, version 0.9.z. Available: http://www.libhand.org/

Silicon Graphics, Khronos Group, *et al.*, "OpenGL (Open Graphics Library)." Available: https://www.opengl.org/

Sturman, D. J. and Zeltzer, D., "A survey of glove-based input," *IEEE Computer Graphics and Applications*, vol. 14, no. 1, pp. 30–39, Jan 1994.

The OGRE Team *et al.*, "OGRE (Object-oriented Graphics Rendering Engine)." Available: https://www.ogre3d.org/

Toolan, T. M. *et al.* University of Rhode Island. "Guide to Writing Your Thesis in LaTeX." Accessed on April 15, 2018. June 2006. Available: https://web.uri.edu/ecbe/thesisguide/

Toolan, T. M. and Tufts, D. W., "Detection and estimation in non-stationary environments," in *Proceedings IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.

Torrence, A., "Martin Newell's original teapot," in *Proceedings of the ACM SIGGRAPH*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. Available: http://doi.acm.org/10.1145/1180098.1180128

Turabian, K. L., *A Manual for Writers of Term Papers, Theses, and Dissertations, 6th. edn.* Chicago, Illinois, United States of America: University of Chicago Press, 1987.

Šarić, M., "Libhand: A library for hand articulation," 2011. Available: http://www.libhand.org/

Wachs, J. P., Kölsch, M., Stern, H., and Edan, Y., "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, Feb. 2011. Available: http://doi.acm.org/10.1145/1897816.1897838

Wang, J.-W., Wang, C.-C., and Lee, J.-S., "Genetic eigenhand selection for hand-shape classification based on compact hand extraction," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2215–2226, 2013. Available: http://www.sciencedirect.com/science/article/pii/S0952197613001218

Wang, R. Y. and Popović, J., "Real-time hand-tracking with a color glove," in *Proceedings of the ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 63:1–63:8. Available: http://doi.acm.org/10.1145/1576246.1531369

Wang, Z., Simoncelli, E. P., and Bovik, A. C., "Multiscale structural similarity for image quality assessment," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, vol. 2, Pacific Grove, CA, USA, Nov. 2003, pp. 1398–1402 Vol.2.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

Wu, M. and Balakrishnan, R., "Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays," in *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '03. New York, NY, USA: ACM, 2003, pp. 193–202. Available: http://doi.acm.org/10.1145/964696.964718

Wu, Y. and Huang, T. S., "Hand modeling, analysis and recognition," *IEEE Signal Processing Magazine*, vol. 18, no. 3, pp. 51–60, May 2001.